5-11-2002

Support Vector Machines for Speech Recognition

Aravind Ganapathiraju

SUPPORT VECTOR MACHINES FOR SPEECH RECOGNITION

By

Aravind Ganapathiraju

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

May 2002

SUPPORT VECTOR MACHINES FOR SPEECH RECOGNITION

By

Aravind Ganapathiraju

Approved:

Joseph Picone Professor of Electrical and Computer Engineering (Director of dissertation)	Lois C. Boggess Professor of Computer Science (Minor Advisor)
Nicolas Younan Professor of Electrical and Computer Engineering (Committee Member)	Robert J. Moorhead Professor of Electrical and Computer Engineering (Committee Member)
James C. Harden Graduate Coordinator of Computer Engineering in the Department of Electrical and Computer Engineering	G. Marshall Molen Department Head of the Department of Electrical and Computer Engineering
A. Wayne Bennett Dean of the College of Engineering	

Name: Aravind Ganapathiraju

Date of Degree: May 11, 2002

Institution: Mississippi State University

Major Field: Computer Engineering

Major Professor: Dr. Joseph Picone

Title of Study: SUPPORT VECTOR MACHINES FOR SPEECH RECOGNITION

Pages in Study: 185

Candidate for Degree of Doctor of Philosophy

Hidden Markov models (HMM) with Gaussian mixture observation densities are the dominant approach in speech recognition. These systems typically use a representational model for acoustic modeling which can often be prone to overfitting and does not translate to improved discrimination. We propose a new paradigm centered on principles of structural risk minimization using a discriminative framework for speech recognition based on support vector machines (SVMs). SVMs have the ability to simultaneously optimize the representational and discriminative ability of the acoustic classifiers. We have developed the first SVM-based large vocabulary speech recognition system that improves performance over traditional HMM-based systems. This hybrid system achieves a state-of-the-art word error rate of 10.6% on a continuous alphadigit task — a 10% improvement relative to an HMM system. On SWITCHBOARD, a large vocabulary task, the system improves performance over a traditional HMM system from 41.6% word error rate to 40.6%. This dissertation discusses several practical issues that arise when SVMs are incorporated into the hybrid system.

DEDICATION

Lead me from the unreal to the real.
Lead me from darkness to light.
Lead me from death to immortality.

- an invocation from the Upanishads

I would like to dedicate this dissertation to my parents and my brother for having supported me through my education and encouraging me to strive for excellence.

ACKNOWLEDGMENTS

It all started the day the syllable research group met with Vapnik during the WS'97 workshop at Johns Hopkins University. We were impressed by the concept of Support Vector Machines but Vapnik was convinced that they wouldn't work for speech. So we thought, "No way, that can't be true", and here I am on the verge of presenting my work on how support vector machines can be used for speech recognition. This would not have been possible without the motivation and mentoring provided by Joe Picone throughout my graduate studies. The long discussions about being a speech researcher and those design meetings for the ISIP speech recognition toolkit have helped me get a better perspective of research in general and speech research in particular. As my advisor and a colleague, I am truly indebted to Joe for making this dissertation a reality.

Being one of the first graduate students of this small speech group from the "Deep South" has made my time in the graduate school even more enjoyable. I remember all those Friday morning philosophy sessions with Jon Hamaker, Neeraj Deshmukh and Joe. I would like to thank the three of them for the thought-provoking discussions, continuous encouragement, and, last but not least, fresh bagels they provided during my research. I would like to thank the rest of the ISIP group for making my years at ISIP memorable.

I would like to thank Thorsten Joachims for providing the SVMLight toolkit which was used for most of the experimentation for my dissertation. He was patient enough to

also personally communicate about some of the issues regarding the data-cleanup properties of support vector machines.

Of course all this would not have been possible without the constant encouragement and support provided by my parents, and my brother and sister throughout my studies. I would like to especially thank my father for encouraging me to strive for excellence and for advising me to pursue what matters the most to me. I would like to thank my wife for supporting me and waiting patiently for two years to finally “tie the knot”. I thank her for listening through all the whining I had to do when my experiments failed or when debugging was not being fun. I would also like to thank my roommate throughout my graduate studies, Udai Kumar, for bearing with my techno-blabber and listening to the frustrations and successes at work.

Finally, I would like to thank the College of Engineering — Mississippi State University, National Science Foundation and the Department of Defense for supporting this research.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Several experiments in understanding the human cognitive process over the years have confirmed that equal-sized physical differences in the signals arriving at our sensory organs are perceived as being smaller within categories and larger between categories [1]. For example, differences in wavelength within the range of the color yellow are perceived as smaller than equal-sized differences in the range between yellow and green [1]. Is a similar phenomenon inherent in the sounds produced and perceived by the human speech system?

Speech is a uniquely human characteristic used as a tool to communicate and express ideas. Automatic speech recognition (ASR) technology has made significant progress over the past few decades. ASR systems have progressed from being able to handle small vocabularies such as digits, to large vocabularies such as broadcast news which can easily reach tens of thousands of words. In recent technology evaluations conducted by the National Institute for Standards and Technology (NIST) [2], the best systems perform at accuracies above 80% on tasks approaching unrestricted transcription.

Humans are very adept at recognizing speech and their performance is stable even under adverse conditions [3,4,5]. ASR systems, however, still fall far short of human performance on conversational speech tasks [4]. In listening tests conducted on a large

vocabulary task, recognition accuracy by humans was found to be an order of magnitude higher than machines (measured in terms of the number of words incorrectly transcribed). Though these tests included data with varied signal qualities, human recognition performance was found to be consistent over a diverse set of conditions.

To a large extent, it appears that humans do speaker and channel adaptation effortlessly. This can be clearly seen in the case of recognizing television broadcasts where the background environment changes significantly — news broadcasts, live interviews, advertisements etc. In all the above cases humans easily adapt to the rapid changes while ASR systems suffer at handling these environment changes [6].

1.1. Complexity of the Speech Recognition Problem

The demand for applications with voice interfaces has recently enjoyed tremendous growth [7] as part of the Internet revolution and extends far beyond dictation or other primitive forms of man-machine communication. Archiving and indexing [8,9] audio data, though not originally envisioned in the early days of speech recognition research, is now one of the fastest growing voice applications. ASR also finds application in electronic devices that are too small to allow data entry via the commonly used input devices such as keyboards. Personal Digital Assistants (PDA) and cellular phones are such examples in which ASR technology is beginning to play an important role.

Though humans seem to effortlessly recognize speech even in adverse environmental conditions, ASR systems significantly lag human performance on challenging tasks such as conversational speech [4]. What makes conversational speech

difficult? Listed below are some of the most significant factors that make the task difficult:

- **Style of Speech**: An important issue when comparing various recognition tasks, and a problem addressed in this dissertation, is the style of speech — read, continuous, spontaneous, disfluent, etc. Early speech research centered around recognizing isolated words which were delimited by pauses between utterances. Continuous speech requires the development of robust segmentation strategies as well as speech/non-speech classification. Conversational speech systems must allow non-speech sounds, interjections, restarts and a slew of other disfluencies [10-14]. Such artifacts increase acoustic confusibility and necessitate improvements in acoustic modeling.
- **Speaker Dependence**: A major application for voice interfaces is database query — users accessing data by issuing voice commands. This is a task humans can do much more easily by voice than by the keyboard, since speech is a natural mode for communication in humans. Modes such as keyboards are not only uncomfortable but also nonintuitive for naive users. These tasks favor voice interfaces that are speaker-independent, especially when interfacing over keyboard-less devices such as the telephone. On the other hand, speech dictation systems available on personal computers are designed to adapt to speaker characteristics, and offer sophisticated customization features. There is a nontrivial increase in complexity when we transition from speaker-dependent to speaker-independent systems. This dissertation deals entirely with the design of speaker-independent systems. Similar to the problems with

conversational speech, speaker independence increases acoustic confusibility by increasing the variability inherent in each acoustic unit, though not to the same degree.

- **Recording Conditions**: It is well known that the performance of a speech recognition system is proportional to the signal-to-noise ratio of the input signal [2,4]. Noise can be introduced into the signal in many ways. It could have been added during the recording process or by the ambient conditions. For example, speech recorded over the telephone is degraded by the bandwidth constraints and non-linearities imposed by the telephone handset. The analog properties of the local loop extending to a customer's premises, and echo also add to the degradation. Further, ambient noise such as other speakers carrying on conversations in the background (referred to as babble noise in the literature) poses a very severe problem for current systems. Cellular telephones and wireless communications networks constitute one of the most challenging environments for speech recognition due to noisy ambient conditions (automobiles and crowded public places), unpredictable networking problems (fading and packet loss), and degradations in the signal introduced by speech compression algorithms. Such problems serve to increase acoustic confusibility, thereby warranting powerful acoustic models.

In summary, improving discrimination in recognition systems is one of the fundamentally important research areas for speech recognition. This is the general area addressed in this dissertation. A demonstration of the importance of discrimination is shown in Fig. 1. A

scatter plot of the first two elements of our standard feature vector for all vowels in the SWITCHBOARD Corpus [15] is shown. The features for each vowel are plotted in a different color. The data includes both male and female adult speakers. We can see that the overlap between the distributions of these features is severe. To succeed in speech recognition, we need to explore acoustic units spanning more than a frame of the input signal and classifiers that can handle potentially high-dimensional data with better acoustic discrimination.

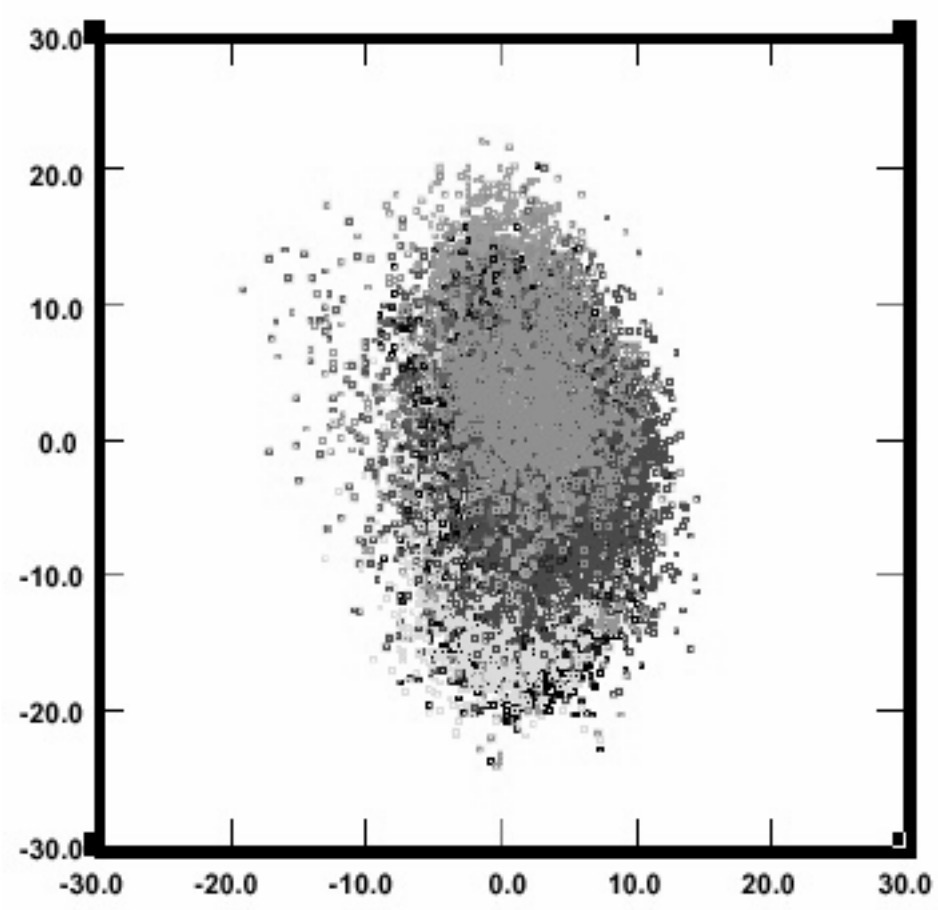

Figure 1. Overlap of the distribution of features for conversational speech is one of the fundamental problems in speech recognition. The first two cepstral coefficients are shown for all vowels in the SWITCHBOARD corpus. (Each color denotes a different vowel.)

Approaches to improving discriminability can be categorized in two ways. In the first category, discrimination is improved by operating the recognizer in a feature space in which the acoustic units of interest are inherently better separated. This acoustic feature space is typically multi-dimensional and data can be transformed to this space via linear or non-linear transformations [16]. Schemes like linear discriminant analysis (LDA), heteroscedastic LDA (HLDA) and independent component analysis (ICA) fall under this category [18]. These methods have had significant success on speech recorded in controlled settings but have had only marginal success (less than 5% relative improvement in word error rate) on conversational speech.

In a second category, the problem of discrimination is addressed at the model level by building better classifiers. Discriminatively-trained hidden Markov models (HMM) and neural networks fall under this category [19-23]. Support Vector Machines (SVMs), a discriminative machine learning technique which is the basis of this dissertation, also falls into this second category [24-30]. A common theme amongst these techniques is the explicit incorporation of a quantity that measures discrimination into the parameter estimation process. Parameter estimation algorithms attempt to maximize a cost function related to this discrimination quantity. This second category has resulted in better performance on a wide range of tasks [19,20,24,31]. For example, HMMs estimated using a Maximum Mutual Information (MMI) criterion achieved significant improvements (about 10% relative) on a standard conversational speech evaluation task [32,33].

1.2. Statistical Speech Recognition

Our approach to deal with the problem previously cited is largely statistical in nature. The goal in a statistically-based speech recognition system is to find the most likely word sequence given the acoustic data. If A is the acoustic evidence that is provided to the system and $W = w_1, w_2, \ldots, w_N$ is a sequence of words, then the recognition system must choose a word string $\hat{W}$ that maximizes the probability that the word string W was spoken given that the acoustic data A was observed:

$$\hat{W} = \underset{W}{\text{argmax}}\ p(W/A) \ . \tag{1}$$

$p(W/A)$ is known as the *a posteriori* probability since it represents the probability of occurrence of a sequence of words after observing the acoustic signal A. The above approach to speech recognition, where the word hypothesis is chosen within a probabilistic framework, is what makes most present recognizers statistical pattern recognition systems.

It is difficult, if not computationally intractable, to directly compute the above maximization since there are effectively an infinite number of word sequences for a given language from which the most likely word sequence needs to be chosen. This problem can be significantly simplified by applying a Bayesian approach to finding $\hat{W}$:

$$\hat{W} = \underset{W}{\text{argmax}}\ p(A/W)p(W) \ . \tag{2}$$

The probability, $P(A/W)$, that the data A was observed if a word sequence W was spoken is typically provided by an *acoustic model*. The likelihood $p(W)$ that gives the *a priori* chances of the word sequence W being spoken is determined using a *language model* [34,35]. Probabilities for word sequences are generated as a product of the acoustic and language model probabilities. The process of combining these two probability scores and sorting through all plausible hypotheses to select the one with the maximum probability, or likelihood score, is called decoding or search. Fig. 2 depicts the above framework under which most ASR systems operate.

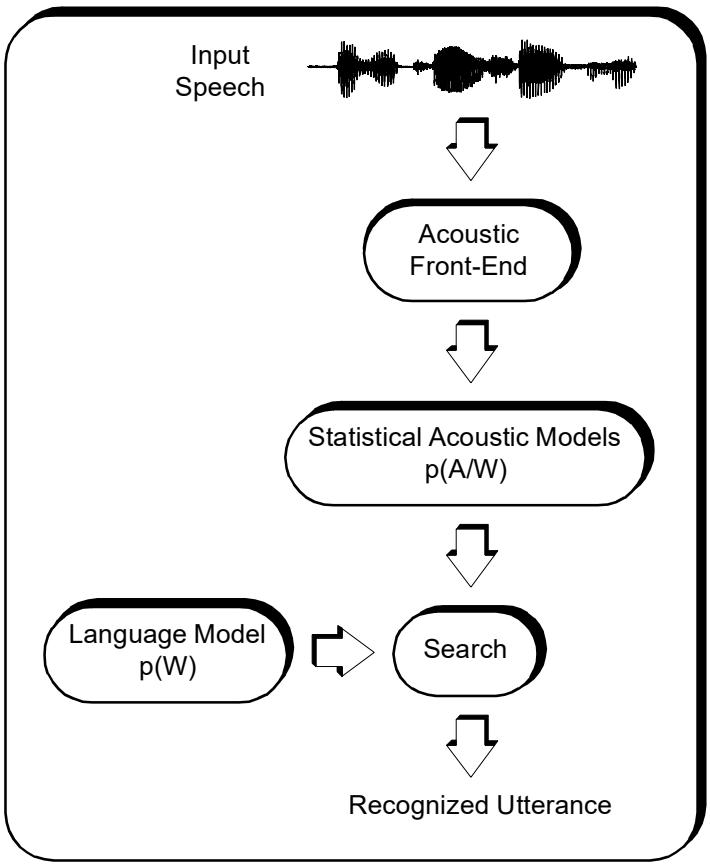

Figure 2. Schematic overview of a statistical speech recognition system.

1.3. Current Acoustic Modeling Technology

In most current speech recognition systems, the acoustic modeling components of the recognizer are almost exclusively based on hidden Markov models (HMMs) [34-39]. The ability to statistically model the variability in speech has been the main reason for the success that HMMs have enjoyed over the years. HMMs provide an elegant statistical framework for modeling speech patterns using a Markov process that can be represented as a state machine. The temporal evolution of speech is modeled by an underlying Markov process [37]. The probability distribution associated with each state in an HMM models the variability which occurs in speech across speakers or even different speech contexts.

The complete description of the model can be provided using the following quantities:

- N — the number of states
- The state-transition probability distribution $\underline{A} = \{a_{ij}\}$
- The output probability distribution $\underline{B} = \{b_j(o)\}$, where o is the input observation vector.

The output probability distribution represents the probability of observing an input feature vector in a given state. At the core of the HMM is a Bayes classifier where classification is done using a simple likelihood ratio test. The output probability distribution could be parametrized in several ways. The distributions could be discrete or continuous. The choice between discrete and continuous distributions depends on several factors including:

- continuous distributions provide more accurate modeling
- trade-off between accuracy of modeling and amount of available training data is an issue for continuous densities
- discrete distributions are typically estimated using some form of vector quantization — vector quantizer becomes a critical component
- probability computation in the case of discrete distributions can be a simple lookup table whereas for continuous distributions it requires a significant amount of computation

The most common form of the probability density used in speech recognition is a Gaussian. Only continuous densities will be considered in this dissertation. The most commonly used form of the output distribution is a multivariate Gaussian distribution[1]. A multivariate Gaussian can be written as:

$$b_j(o_t) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_j|}} \exp\left(-\frac{1}{2}(o_t - \mu_j)'\Sigma_j^{-1}(o_t - \mu_j)\right) , \qquad (3)$$

1. Other distributions, such as Laplacians, have been used, but have had limited success [40].

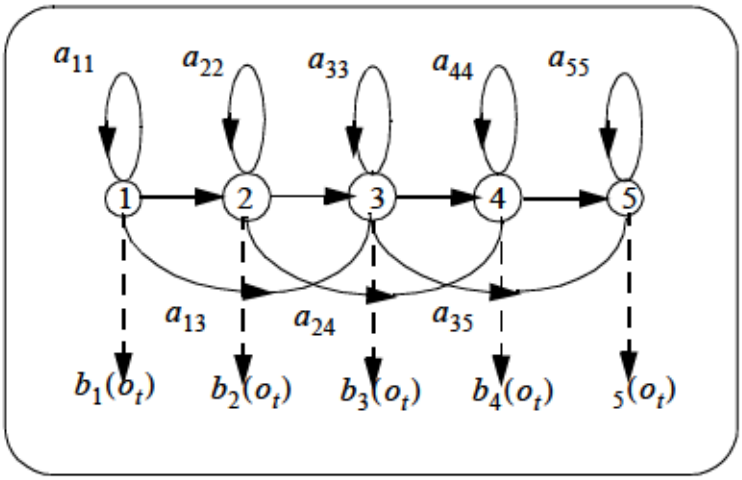

Figure 3. A simple HMM featuring a five state topology with skip transitions.

where n is the dimension of the observation vector $\boldsymbol{o}_t$ at time t and the subscript j indicates that the Gaussian under consideration belongs to the jth state of the HMM. The covariance may be either a full $n \times n$ matrix modeling the correlation between the elements, or can be a diagonal matrix where the dimensions of the feature vector are assumed to be independent. Most modern speech recognition systems use a diagonal covariance matrix. Fig. 3 shows an example of a five state HMM with skip transitions.

Equation (3) when transformed to the logarithm domain is equivalent to a distance metric between the test vector and the mean of the Gaussian modeling the HMM state, commonly known as the Mahalanobis distance [41]. The Mahalanobis distance plays a vital role in the algorithms used for finding the most likely word hypothesis given the acoustic data. Classification into various acoustic units of recognition is done via a direct comparison of Mahalanobis distances of the test vector from the competing acoustic models [41].

1.4. Acoustic Model Estimation

The estimation of the parameters of the acoustic models (like HMMs) plays a vital role in the accuracy of the ASR system. A key to the widespread use of HMMs to model speech can be attributed to the availability of efficient parameter estimation procedures [39]. Maximum likelihood (ML) is one such optimization criterion. The motivation to use ML comes from the probabilistic definition of the speech recognition process which attempts to find a word sequence which maximizes a cost (likelihood) function as described in a previous section.

At the acoustic level, the goal of the recognizer is to classify speech into words (or sub-word units as the case may be). When the acoustic units are modeled using HMMs as defined previously, the probability distributions used to model the observation vectors are used for classification. Since the direct maximization in (1) is not practical, the Bayesian approach is used where the problem is modified to the optimization shown in (2). The parameters of the HMMs need to be estimated using a suitable optimization criterion. The Expectation-Maximization (EM) algorithm provides an iterative framework for ML

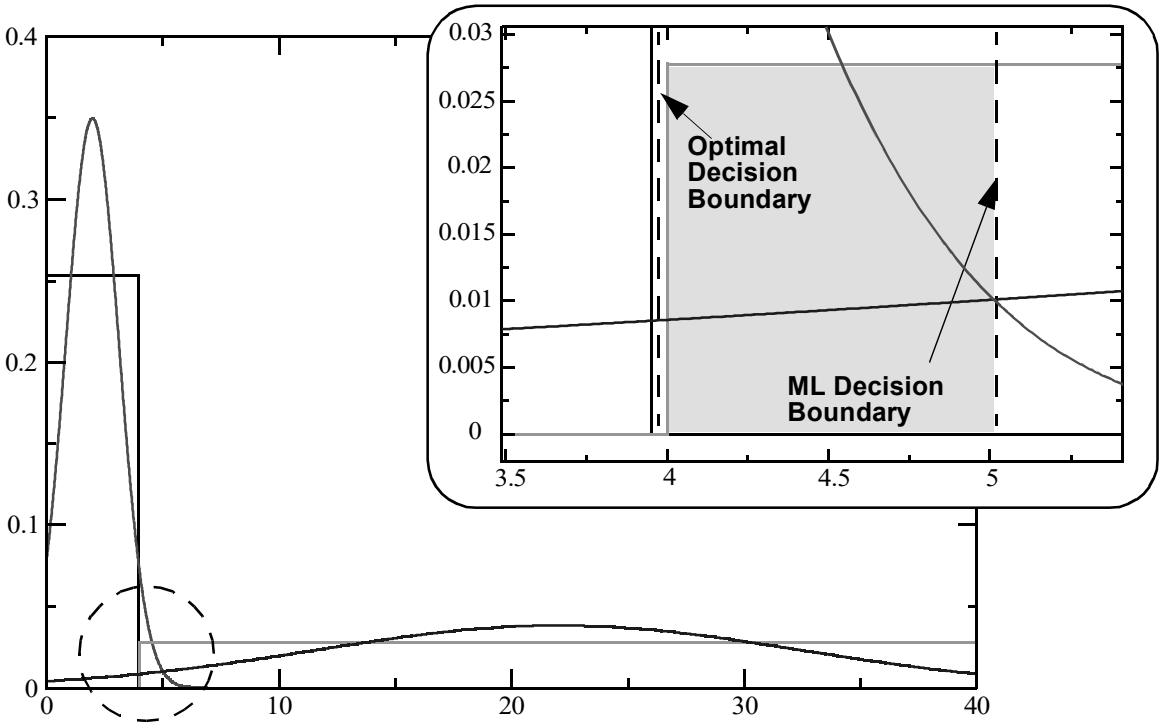

Figure 4. An example of a two-class problem when maximum likelihood decision surface is not optimal (adapted from [31]). In the exploded view, the shaded region indicates the error induced by modeling the separable data by Gaussians estimated using maximum likelihood. This case is common for data, such as speech, where there is overlap in the feature space or where class boundaries are adjacent.

estimation with good convergence properties, though it does not guarantee finding the global maximum [42-44].

There are however problems with ML as formulated above. These include:

- maximizing likelihood does not necessarily translate to better classifiers
- maximum likelihood improves the classifier's ability to represent a specific class but what is really needed is a classifier that better discriminates a class from the others or mathematically, we need to better model $P(W/A)$
- ML estimation is not discriminative in that model parameters are estimated based on in-class data alone without considering the out-of-class data
- ML assumes the form of underlying probabilistic model (typically a Gaussian is used)

Fig. 4 shows a simulated classification problem where using a ML approach does not yield an optimal classifier. The two classes are derived from completely separable uniform distributions. ML is used to fit Gaussians to these classes and a simple Bayes classifier is built. However, we see that the decision threshold occurs inside the range of class 2. This means that the probability of error is significant. However if we were to simply recognize that the range of data points in class 1 is less than 3.3 and that no data point in class 2 occurs within this range, we can achieve perfect classification. In this example any amount of effort expended in learning a better Gaussian will not help achieve perfect classification. More dramatic examples can be constructed to show that learning decision regions discriminatively will help improve classification. The conclusion from the above example is not necessarily that using a Gaussian is an incorrect choice. However, ignoring information about the out-of-class data is definitely a problem.

There has been significant work in the area of discriminative techniques for the estimation of HMM parameters. The primary difference between HMM parameter estimation via ML and discriminative techniques is that the optimization process is provided with negative examples or *out-of-class* data [31]. The foundation for a particular discriminative technique is either based on some information theoretic concept as in MMI estimation or based on directly minimizing the classification error as in Minimum Classification Error (MCE) estimation [31]. Though the above approaches have had significant success in terms of improvements in recognition performance their use has been limited because they require immense resources [32].

Neural networks have also been applied to speech recognition owing to several advantages they offer over the typical HMM systems [45-50]. Neural networks can learn very complex non-linear decision surfaces effectively and in a discriminative fashion. However, their estimation process is significantly more computationally expensive than HMMs and they are typically formulated as classifiers of static data. This has led to the development of several connectionist approaches where the neural networks are embedded in a HMM framework [51,52,53]. The performance of these hybrid systems have been competitive with many HMM-based systems and typically require a significantly reduced parameter count [51]. The hybrid connectionist systems also provide a way to mitigate some of the assumptions made in HMM systems that we know are incorrect for the human speech process [54]. One such significant assumption is that of independence of observations across frames [55-59]. Hybrid systems mitigate this problem by allowing the neural network classifiers to classify based on several frames of

acoustic data at a time [53]. A similar approach will be pursued in this dissertation by processing multi-frame data.

Although the final system configurations for connectionist systems are simpler than HMM based systems, their use has been limited because of various limitations listed below:

- Generalization: Neural networks have been known to overfit data unless specific measures are taken to avoid that. These measures typically include some form of cross-validation which can be restrictive when the amount of training data is limited to start with.

- Optimization Process: Neural network learning is based on the principle of empirical risk minimization via the back-propagation algorithm [60,61,62]. Though this guarantees good performance on the training data, obtaining a bound on the performance on the test data is not easy.

- Model Topology: In most connectionist hybrid systems the topology of the neural network classifiers needs to be fixed prior to the estimation process. This is not always easy without expert knowledge of the data. Techniques do however exist to learn connections automatically but are expensive [63,64].

- Convergence: Convergence of the optimization process has been the biggest drawback of neural networks. Convergence is typically an order of magnitude slower than ML estimation of HMM parameters. Both ML estimation using the EM algorithm and estimation of parameters of the neural networks do not guarantee reaching a global maximum unless measures are taken to perturb the system from time to time which increases the possibility of reaching the global maximum [66,67].

The need for discrimination and classifiers with good generalization and convergence properties that can be used for speech recognition has led us to look at a new machine learning paradigm called the support vector machines (SVM) which forms the basis of this dissertation [68].

1.5. Support Vector Machines

Some of the generalization properties of neural networks have been mentioned in the previous section. Why is generalization important? HMM-based speech recognition systems perform very well on closed-loop tests but performance degrades significantly on open-loop tests [65]. The performance of systems on speaker-dependent tasks is significantly better than on speaker-independent tasks. This can be attributed to the fact that most systems do not generalize well. There is a definite need for systems with good generalization properties where the worst-case performance on a given test set can be bounded as part of the training process without having to actually test the system [68]. With many real-world applications where open-loop testing is required, the significance of generalization is further amplified.

As mentioned in a previous section, empirical risk minimization is one of the most commonly used optimization criteria to estimate classifiers. However, there can be several

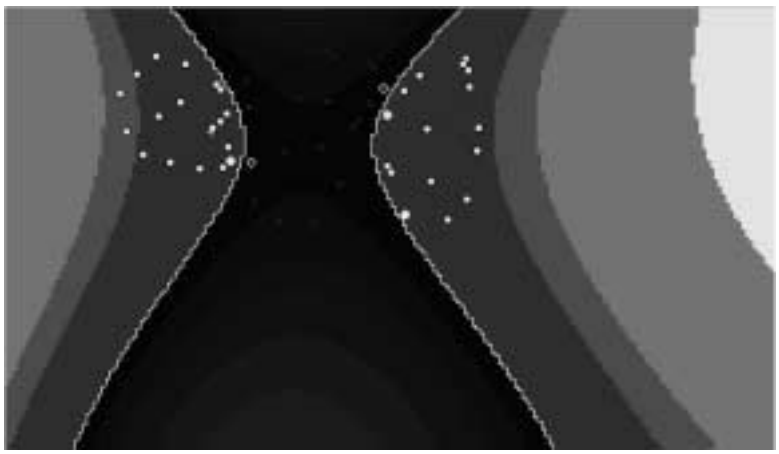

Figure 5. Sample classification by the Royal Holloway SVM applet using a polynomial kernel. This data cannot be classified by a linear separating margin. This is an interesting example in the sense that SVMs can handle such multi-modal data effectively.

configurations of the classifier that can achieve minimum risk on the training set. This is one of the reasons why neural networks can get stuck in local saddle points. The problem then is to decide on the configuration that has the least upper bound on the expected test set error. This is the principle of structural risk minimization (SRM). Support vector machines are founded on this principle and the result of SRM is a classifier with the least expected risk on the test set and hence good generalization [68].

SVMs in their simplest form are hyperplane classifiers. The power of SVMs lies in their ability to implicitly transform data to a high dimensional space and to construct a linear binary classifier in this high dimensional space. Since this is done implicitly, without having to perform any computations in the high dimensional space, neither the dimensionality of the data nor the sparsity of data in the high-dimensional space is a problem with SVMs. The hyperplanes in the high-dimensional transform space result in complex decision surfaces in the input data space as depicted in Fig. 5.

SVMs have been applied successfully on several kinds of classification problems and have consistently performed better than other non-linear classifiers like neural networks and mixtures of Gaussians [50,70]. The dataset that propelled SVMs to prominence in the early 90's was the US Postal Service digit data on which the SVMs achieved the best numbers reported [69]. The development of efficient optimization schemes led to the use of SVMs for classification of larger tasks like text-categorization [70,71].

There were some initial efforts to apply SVMs to speaker recognition in the early 90's [72,73]. This effort had limited success because of the lack of efficient

implementations of the SVM estimation process at that time. SVMs have also been applied to simple phone classification tasks and the results have been very encouraging [73-75]. Notice however that all the above classification tasks have one common feature — these are all static classification tasks. SVMs are not designed to handle temporal structure of data. Speech however evolves with time and we need to address this problem in order to harness the advantages of SVMs for speech recognition. This is the primary contribution of this dissertation wherein we have developed a hybrid SVM/HMM framework with the HMM structure being used to handle the temporal evolution of speech and SVMs being used to discriminatively classify frames of speech. The end result is a first successful application of SVMs to continuous speech recognition [73,76,77].

1.6. Dissertation Contributions

The primary objective of this dissertation is to explore the use of SVMs for speech recognition. SVMs have had significant success in the past few years on a wide range of classification problems but have not been applied to the speech recognition problem. Speech recognition, especially continuous speech, suffers from significant acoustic confusibility and discriminative techniques have been successful at dealing with this problem. This dissertation describes the first successful application of SVMs to speech data at various levels of granularity — starting with simple frame level phone classification all the way up to a complete ASR system for continuous speech.

A significant contribution of this dissertation is the hybrid SVM/HMM framework that has been developed to apply SVMs to speech recognition. Though the hybrid

framework has been used in other related areas, several issues specific to SVMs have been addressed in this dissertation. Some of the other specific contributions of this work include:

- SVM distance to likelihood mapping: Several schemes have been studied to convert SVM distances to likelihoods in order to fit the SVM classifiers into the HMM-based ASR system. The sigmoid-based warping function has been found to be sufficiently accurate.
- Segment-level data: Segment level data has been used with the hybrid system developed in this dissertation. Issues related to the generation of this segmental data have been addressed in this dissertation.
- N-best list generation: The hybrid system operates on N-best lists generated from the baseline HMM system. As part of this dissertation, the N-best list generation capability has been added to the ISIP ASR toolkit.
- Fisher Score-based features: The theory for the application of Fisher Scores as input features for the hybrid system with the aim of improving discrimination has been developed.

1.7. Structure of the dissertation

Chapter 2 of this dissertation provides background information on HMM technology as applied to speech recognition. Specifically the details of acoustic modeling in the baseline ASR system are discussed. Chapter 3 describes discriminative training techniques for HMMs that have been successfully applied to continuous speech in the recent past. These techniques include MMI and MCE. Chapter 3 also includes a brief description of neural network based approaches and details hybrid connectionist systems due to their relevance to the hybrid system developed in this dissertation.

Chapter 4 describes the core technology investigated in this dissertation, Support Vector Machines. The chapter includes detailed mathematical formulation of SVMs and

their estimation. Chapter 4 also describes the mathematics behind the use of Fisher scores for speech recognition which help closely integrate HMMs with SVMs. Chapter 5 describes several aspects of the hybrid SVM/HMM speech recognition system that was developed as part of this dissertation. Implementation details of the hybrid ASR system are provided.

Chapter 6 describes the corpora that are used for evaluating the hybrid system. Chapter 7 discusses the core set experiments used to validate the usefulness of the technology described and developed in this dissertation. Chapter 8 concludes the dissertation with discussion of promising avenues of work to continue development of the technology that uses SVMs more effectively for speech recognition.

CHAPTER 2

MAXIMUM LIKELIHOOD-BASED ACOUSTIC MODELING

The most common model of speech production is based on the assumption of separate and independent source and vocal tract models. The speech signal can then be modeled as the convolution of an excitation signal and the vocal tract. Most recognition systems model the vocal tract characteristics only via homomorphic transforms [16]. The vocal tract shape can be modeled using tube models or using statistical methods [34].

Most current automatic speech recognition (ASR) systems employ statistical models to model speech (in reality only the vocal tract characteristics). In the past, systems incorporated a template-matching mechanism based on dynamic programming principles to match the incoming speech to representative patterns, or templates, of the speech units being modeled [34]. However, in recent years, systems have greatly benefited from the ability of statistical approaches to model variability, including speaker and channel variability. Model parameters in these systems are estimated using a data-driven framework involving large amounts of data. A popular data-driven parameter estimation technique is based on the Maximum Likelihood (ML) principle.

This chapter introduces the motivation for ML-based parameter estimation in typical ASR systems. The acoustic modeling component of the system is described more rigorously so as to clearly establish the dependence of contemporary approaches on ML

techniques. There are several issues that arise in the practical implementation of the ML estimation procedure that make the process extremely computationally expensive. In this chapter we will discuss these issues in the context of a typical ASR system [78,79]. The estimation procedure for HMMs is a key starting point for this dissertation since the one goal of this work is to develop an ASR system which is a hybrid between the HMM technology and SVMs. Since complexity is a major issue in implementation of such systems, a detailed analysis of the complexity of the ML approach is presented.

2.1. Acoustic Front-end

The success of a recognition system when posed as a statistical pattern recognition system depends on both the efficacy of the classifiers as well as the space in which the classifiers operate. The acoustic front-end in a speech recognition system transforms raw speech data into a feature space where the acoustic units of recognitions are classified. The feature space into which the speech data is transformed needs to exhibit some important properties.

- Class Separability: Since the goal of speech recognition is the accurate classification of the speech sounds, the basic units of recognition need to be well separated in this feature space. The frequency domain is typically the preferred space [16] in which the classifiers operate.
- Compact Representation: The feature space needs to have a more compact representation than the original input space. This allows for the design of classifiers of moderate complexity and makes accurate parameter estimation possible. Most ASR systems [38] operate in a 30-50 dimensional feature space.
- Knowledge-Based Representations: The feature space in which the classifiers operate needs to support application of *a priori* knowledge about the human speech processing system. For example, nonlinear frequency scale warping, which is used in most recognition systems today [16], is motivated by the logarithmic response of the human auditory system.

Cepstral analysis is by far the most commonly used feature extraction model [16,17]. The motivation for performing analysis in the cepstral domain is two-fold. First, the speech signal contains two key pieces of information: the excitation and the vocal tract shape. Recognition systems typically operate on the latter while speaker identification systems model the former. Cepstral analysis, a form of homomorphic signal processing, provides a mechanism for separating out these two components of the signal. Second, cepstral processing is attractive because it gives the ability to improve noise robustness via very simple and inexpensive mechanisms (such as cepstral mean normalization [5]).

The idea of a *cepstrum* is based on a goal of separating two signals by deconvolution. Speech can be modeled [16] as the convolution of the excitation signal, $e(n)$, and a vocal tract impulse response, $v(n)$:

$$s(n) = e(n) \otimes v(n). \tag{4}$$

Convolution in the time domain is equivalent to multiplication in the frequency domain:

$$S(f) = E(f)V(f) \quad . \tag{5}$$

Taking the log of both sides,

$$\log(S(f)) = \log(E(f)) + \log(V(f)) \quad , \tag{6}$$

The log frequency domain representation of the input signal, $\log(S(f))$, is referred to as the log spectrum. It is (theoretically) possible to remove the contribution of the excitation signal using simple techniques such as spectral subtraction [80]. In practice, a simple Fourier transform can be used to compute the log spectrum.

The time-domain representation of the log spectrum, which is defined as the *cepstrum*, is obtained using an inverse Fourier transform:

$$c(n) = \frac{1}{N_s} \sum^{N_s} \log\left|S_{avg}(k)\right| e^{j\frac{2\pi}{N_s}kn} \qquad 0 \le n \le N_s - 1 \qquad (7)$$

Typically only the lower order terms of the cepstrum are used for speech processing since they represent the short-term correlations that exist because of the influence of the vocal tract shape on the signal. The cepstrum can be computed without having to transform the data into the frequency domain using Linear Prediction (LPC) coefficients [16]. Incorporating the warped frequency scale is simple when the spectrum is available. With LPC-based cepstral computation warping the frequency scale is computationally more expensive and is achieved using a bilinear transform[16]. Finally, the cepstral domain does indeed satisfy several requirements of a "good" feature space [16] for speech processing. Several other feature spaces motivated by our knowledge of the speech signal and our desire to be tolerant of ambient noise [5] have recently shown some promise.

2.2. Parametric vs. Non-Parametric Modeling

Mathematical models of the underlying units that we assume to compose a speech signal are called *acoustic models*. Typical units used to model speech signals are phones, syllables, and words [81-85]. An acoustic model, which is a classifier, can either be a parametric representation of the acoustic unit or a non-parametric representation. HMMs, for example, are parametric classifiers while classifiers based on k-means clustering [41]

are non-parametric. Template matching, as described in the introduction, is an example of a parametric model of speech since it models speech using averages of exemplars of the units. Modeling based directly on exemplars suffers from several drawbacks including:

- Finding the best exemplars is very time-consuming and requires expert knowledge in acoustic phonetics.
- Context-dependent modeling of speech units, which is a critical component of modern technology, is extremely subjective.
- Heuristics have to be applied to achieve accurate temporal modeling (such as duration penalties).
- Modeling speaker variability and background environment is expensive since it is typically done explicitly using multiple models.

In pattern recognition it is very rare that we have complete knowledge of the data that we process. We do however have knowledge of some trends. Parametric models are typically a result of incorporating this partial knowledge into a closed-form representation. Our goal is to use a sufficiently general model that can learn the underlying structure. A Gaussian mixture model [34] is a good example of a sufficiently general parametric model. On the other hand, non-parametric models are attractive because they can automatically understand and accommodate unforeseen modalities in the data without having to assume the properties of an underlying distribution.

There are several aspects of the classification problem that need to be considered before we make a decision on the form of the solution:

- Generalization: In most problems we only have access to a limited amount of data. We need the model to capture all variability in the data in a way that is consistent with unseen data that shares some common statistical properties. Parametric models can be estimated with controlled generalization capabilities.

Most non-parametric solutions have trouble with generalization, and must be constrained using significant *a priori* knowledge about the problem.

- Classification: A classification scheme must also be added to a parametric model to form a pattern recognition system. However, with non-parametric techniques, we can skip the estimation of the data distribution and directly estimate the classifier (or the decision region). HMMs use a parametric form of classification while SVMs are non-parametric.

- Compactness: Most real world systems have access to limited computing resources and limited amounts of training data. This makes the compactness of representation of the classifier an important aspect to consider. For example, modeling a probability density as a Gaussian requires estimation of a minimal number of parameters — the mean and the variance. Comparable non-parametric representations, typically based on vector quantization [85] techniques, would require a significantly greater number of parameters.

Fig. 6 shows the distribution of the 5^{th} cepstral coefficient for the male speakers of the TIDIGITS [87] speech corpus. A Gaussian fit to this data reveals a fairly close match. On the other hand, in Fig. 7, we present a distribution for the 3^{rd} filter bank

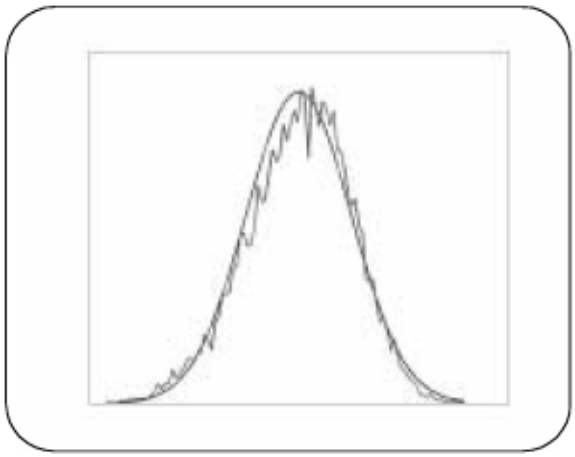

Figure 6. Distribution of the 5th cepstral coefficient for male speakers in the TIDIGITS data.

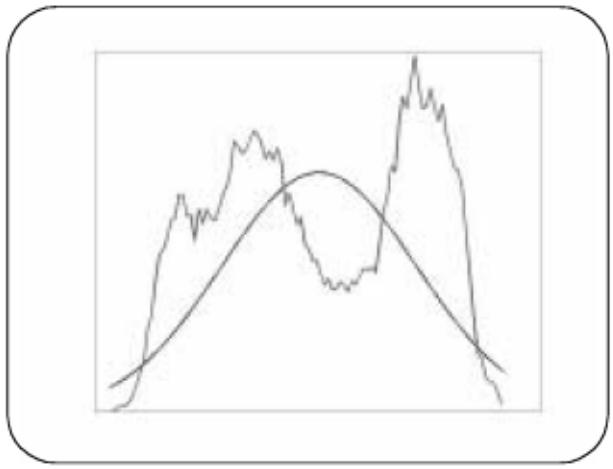

Figure 7. Bimodal distribution exhibited by a specific spectral bin. Note the amount of mismatch in modeling this data with a single Gaussian probability density function.

amplitude [16] for a subset of male speakers from the TIDIGITS speech corpus. Clearly, this feature is multimodal. Speech signal features often demonstrate modalities due to the gender of the speaker, variations in microphones, acoustic channels, and dialect. Such data cannot be modeled by a single Gaussian distribution. A mixture of Gaussians is used to handle the multimodal nature exhibited by most feature spaces. The output probability distribution can be expressed as a mixture of Gaussians by,

$$b_j(\boldsymbol{o}_t) = \sum_{K} c_{jk} b_{jk}(\boldsymbol{o}_t) , \tag{8}$$

where K is the number of component densities in the mixture, c_{jk} is the mixture weight (with the constraints $0 \le c_{jk} \le 1$ and $\sum_{k=1}^{K} c_{jk} = 1$) and b_{jk} is a multivariate Gaussian density function as defined in (3).

2.3. Estimation-Maximization and Maximum Likelihood

If we assume that parameters of an HMM are fixed but unknown, we can pose the problem of parameter estimation as one that maximizes the probability that the model generates the observed data (*a posteriori* probability). The approach for a typical HMM parameter estimation process then becomes maximization of the likelihood of the data given the model, traditionally known as Maximum Likelihood (ML) estimation [39]. One of the most compelling reasons for the success of ML and HMMs has been the existence of iterative methods to estimate the parameters while guaranteeing convergence.

Expectation-Maximization (EM) is one algorithm that is used extensively to perform ML estimation [42-44].

Suppose we have a feature space Y of "complete data" — the data is complete in that there is no missing information. Suppose we are given a measurable map $y \to x$ of Y to a measurable space X of "incomplete data" — the data is incomplete because there is missing information due to the lack of knowledge of the underlying processes or actual loss of information. The "incomplete data" space is typically what we have access to in real world problems. Let $f(\boldsymbol{y}|\Phi)$ be a member of the set of probability density functions defined on Y and let $g(\boldsymbol{x}|\Phi)$ be the density function on X induced by $f(\boldsymbol{y}|\Phi)$. The symbol Φ represents the parametrization of the density function (e.g., the mean and the variance for a Gaussian model).

For a given $\boldsymbol{x} \in X$ the goal of EM is to find the maximum of the log likelihood function $L(\Phi) = \log(g(\boldsymbol{x}|\Phi))$ by using the relationship between f and g. In other words, instead of optimizing directly in the observation space, X, we attempt to optimize in the "complete" space, Y, and use the relationship between the two spaces to guarantee an optimal solution. EM is very effective for problems where maximizing based on f is significantly easier than maximizing based on g. In reality, we need not necessarily perform the optimization in the logarithm domain. Given that the most common parameterizations of the density functions are exponential in nature, the logarithm domain simplifies the mathematical analysis significantly.

The distributions, f and g are related through the conditional distribution k on $Y(\boldsymbol{x})$:

$$f(\boldsymbol{y}|\Phi) = k(\boldsymbol{y}|\boldsymbol{x}, \Phi)g(\boldsymbol{x}|\Phi). \quad (9)$$

We can then write the relationship between the log likelihoods as,

$$E(\log f(y|\Phi)|x, \Phi') = E(\log(k(y|x, \Phi))|x, \Phi') + E(\log(g(x|\Phi))|x, \Phi') \ , \quad (10)$$

or,

$$E(\log(g(x|\Phi))|x, \Phi') = E(\log f(y|\Phi)|x, \Phi') - E(\log(k(y|x, \Phi))|x, \Phi'), \quad (11)$$

or,

$$L(\Phi) = Q(\Phi|\Phi') - H(\Phi|\Phi'), \quad (12)$$

where $Q(\Phi|\Phi')=E(\log f(\boldsymbol{y}|\Phi)|\boldsymbol{x}, \Phi')$ and $H(\Phi|\Phi')=E(\log k(\boldsymbol{y}|\boldsymbol{x}, \Phi)|\boldsymbol{x}, \Phi')$ are the auxiliary functions.

The EM formulation then can be implemented as a two-step process. Step 1 or the E-step includes the determination of $Q(\Phi|\Phi^c)$ where Φ^c is the parameter set for the current iteration. Since this computation involves f, it may have a simple closed-form definition. On the other hand, for problems that do not have closed-form solutions, there are a number of iterative numerical solutions [88] that work quite well. Step 2, or the M-step, involves maximizing the auxiliary function $Q(\Phi|\Phi^c)$ and choosing the parameter set Φ^* that maximizes this function. This parameter set is then used as the

value for the E-step of the next iteration. This iterative process is guaranteed to make L, the log-likelihood function, monotonically increasing [89]. The M-step guarantees that

$$Q(\Phi^{*}\left|\Phi^{c}\right.) \geq Q(\Phi^{c}\left|\Phi^{c}\right.). \tag{13}$$

From Jensen's inequality [42] it can be shown that

$$H(\Phi^{*}\left|\Phi^{c}\right.) \leq H(\Phi^{c}\left|\Phi^{c}\right.) \ . \tag{14}$$

Equations (13) and (14) imply that, if we can guarantee the auxiliary function Q to satisfy (13), then

$$L(\Phi^{*}) \geq L(\Phi^{\mathrm{c}}), \tag{15}$$

and the iterative process will ultimately lead to the *local* maximum likelihood solution. The EM formulation does not guarantee a global optimal solution will be identified.

The EM algorithm has become an extremely important technique for parameter estimation for two reasons. First, EM can be used successfully when the optimization of the auxiliary function is simpler than optimization of the primary function. Second, if there is missing data involved in the optimization process, EM can elegantly arrive at the optimal solution without having to explicitly estimate the missing data. Since real world problems typically involve insufficient data, this feature of EM becomes extremely important in practice.

2.4. HMM Parameter Estimation

As described in the previous section, the power of EM-based ML estimation lies in the definition of the auxiliary function. The easier it is to evaluate and optimize the auxiliary function the faster we can compute the ML estimates of the parameters of interest. The goal of HMM parameter estimation is to maximize the likelihood of the data under the given parameter setting. In an iterative framework this can be written as,

$$P_{\Phi}(y) > P_{\Phi'}(y), \tag{16}$$

where Φ' is the parameter setting for the previous iteration and Φ is the current parameter setting. However, in reality, evaluating and optimizing the density function can be a lost cause both in terms of computational resources and accuracy because we have to make a serious assumption about the data at hand — that is, that we have access to an infinite amount of data generated according to the distribution $P(y)$.

We can alternatively define an auxiliary function such that we can guarantee an optimal solution with the help of the EM theorem. If,

$$\sum_{t} P_{\Phi'}(s|y)\log P_{\Phi}(s|y) > \sum_{t} P_{\Phi'}(s|y)\log P_{\Phi'}(s|y) \tag{17}$$

then (16) is guaranteed and EM estimation will converge.

The gist of the above formulation is that by starting with a model Φ' and finding a model Φ such that (17) is satisfied, then the observed data y is more probable under the model Φ than under Φ' [39]. In the above formulation s is the intermediate random

variable that depends on the model parameter settings. For example, s could be the state sequence in an HMM, which is not something we directly observed. The terms on the LHS and RHS of (17) can be represented as the auxiliary functions $Q(\Phi', \Phi)$ and $Q(\Phi', \Phi')$ respectively. Since we are maximizing the auxiliary function in the EM framework, the optimal parameter setting can be obtained using simple gradient descent techniques. The parameter update equations can be obtained by differentiating $Q(\Phi', \Phi)$ with respect to each of the parameters and setting the derivative to zero. When s is chosen as the state sequence, the EM formulation is called the Baum-Welch algorithm [89].

2.5. Reestimation Formulae

Before we derive the HMM parameter reestimation equations for use with the EM algorithm, a few probabilities need to be defined.

Forward Probability

The forward probability gives us the probability of generating the observations from time 1 to t and that the state j is visited at time t.

$$\alpha_j(t) = Pr(\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_t, \theta_t = j) \tag{18}$$

The above computation can be efficiently done using the following recursive formulation:

$$\alpha_j(0) = \begin{matrix} 1 & \text{for } j=1 \\ 0 & \text{for } 1<j<N \end{matrix} \tag{19}$$

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_i(t-1) a_{ij} \right] b_j(\boldsymbol{o}_t), \text{ for } 1 \le t \le T \text{ and } 1 < j < N, \quad (20)$$

$$\alpha_N(T) = \sum_{i=2}^{N-1} \alpha_i(T) a_{iN}, \quad (21)$$

where N is the index of the final HMM state and T is the duration of the speech data measured in frames.

Backward Probability

The backward probability is the probability of generating the observations from time $t+1$ to T if the model was in state j at time t.

$$\beta_j(t) = Pr(\boldsymbol{o}_{t+1}, \boldsymbol{o}_{t+2}, \ldots, \boldsymbol{o}_T | \theta_t = j) \quad (22)$$

Similar to the forward probability computation, a recursive formulation exists for the backward probability computation.

$$\beta_i(T) = \begin{matrix} 1 \text{ for i=N} \\ a_{iN} \text{ for 1<i<N'} \end{matrix} \quad (23)$$

and,

$$\beta_i(t) = \sum_{i=2}^{N-1} a_{ij} b_j(\boldsymbol{o}_{t+1}) \beta_j(t+1), \text{ for } 1 \le t < T \text{ and } 1 < i < N, \quad (24)$$

Utterance Likelihood

The total probability P of a sequence of observations $\boldsymbol{O}$ given the model M can be written in terms of α and β as:

$$P(\boldsymbol{O}|M) = \sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_j(t)\beta_j(t) \ . \tag{25}$$

This computation can also be efficiently executed using the backward probability:

$$P(\boldsymbol{O}|M) = \beta_1(0), \tag{26}$$

where,

$$\beta_1(0) = \sum_{i=2}^{N-1} a_{1j} b_j(\boldsymbol{o}_1)\beta_j(1). \tag{27}$$

Auxiliary Function

Another fundamental quantity for the estimation process is the probability of observing the sequence $\boldsymbol{O}$ and taking a transition from the i^{th} state to the j^{th} state at time t,

$$P(\boldsymbol{O}, \theta_{t-1} = i, \theta_t = j) = \alpha_i(t-1)a_{ij}b_j(\boldsymbol{o}_t)\beta_j(t) \ . \tag{28}$$

Let θ be a fixed state sequence through the HMM. Then the auxiliary function can be defined as,

$$Q(\Phi, \Phi') = \sum_{\theta \in \Theta} P_{\Phi'}(\boldsymbol{O}, \theta) \log P_{\Phi}(\boldsymbol{O}, \theta), \tag{29}$$

where Θ is the set of all possible state sequences through the HMM and, Φ' and Φ represent old and new parametrization of the HMM respectively.

From the EM formulation in (17), we know that $Q(\Phi, \Phi') > Q(\Phi', \Phi')$ implies that $P_{\Phi}(\boldsymbol{O}) > P_{\Phi'}(\boldsymbol{O})$ — the criterion for maximum likelihood estimation. We also defined

$$P_{\Phi}(\boldsymbol{O}, \theta) = \prod_{t=1}^{T+1} a_{\theta_{t-1}\theta_t} \cdot \sum_{t=1}^{T} b_{\theta_t}(\boldsymbol{o}_t) \tag{30}$$

where θ_t is the state occupied at time t in the sequence θ. Using (30) in (29) and maximizing with respect to each of the HMM parameters, the required parameter reestimation formulae can be derived [19]. A complete derivation of the reestimation procedure can be found in [19] and [39].

2.6. Practical Parameter Estimation

Acoustic modeling, in simple terms, involves developing models that represent the acoustic units which form the fundamental building blocks of spoken language. These acoustic units could be words themselves or sub-word units like phones and syllables. In either case, in an HMM-based ASR system, each acoustic unit is modeled using an HMM. The training process involves the estimation of the parameters of the HMMs given the

speech data. In a simple phone-based system the aim of the training process is to estimate the parameters of the HMMs representing the phones. However the complexity of recognition tasks warrant several improvements to this rather simplistic framework.

2.6.1. Context Dependent Acoustic Models

In tasks where the words are spoken in isolation, the boundary phones of words are not affected by the preceding or the following words. However, in continuous speech, coarticulation is an important phenomenon that occurs within words as well as across word boundaries. To account for coarticulation, context-dependency across words has been successfully modeled using lexical tree-based approaches [90]. When the context of a context-dependent phone spans a word boundary, it is called a *cross-word* model. Otherwise it is referred to as a *word-internal* model. This difference is illustrated in Fig. 8 via an example.

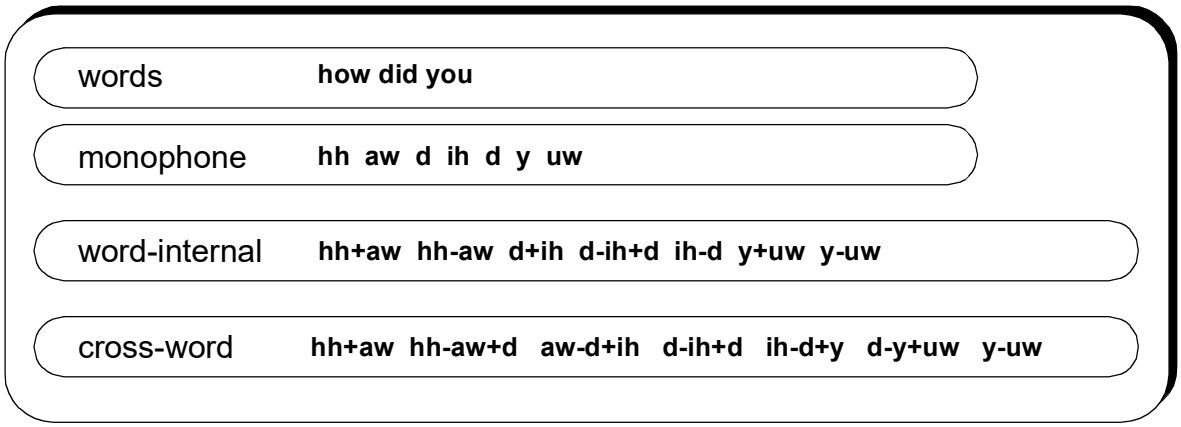

Figure 8. Example of context-dependent phone realization — "+" denotes right context and "-" denotes left-context

2.6.2. Parameter Sharing

Though cross-word context modeling is very effective for continuous speech (in terms of improved accuracy), cross-word modeling is very expensive. First, the number of trainable parameters increases dramatically. Assuming a phone set of size 40, we will typically use a context size of one phone to the left and right of the current phone. Hence, the number of possible context-dependent models in the system could be 40^3. If we use 12 mixture components per state and a 39-dimensional feature vector, the number of trainable parameters approaches $(39+39)\times 12\times 40^3 = 60x10^6$ assuming a diagonal covariance matrix for each Gaussian mixture component. In addition to the computational impact of this, there is also a significant increase in the resources required to manage the acoustic models (disk space and memory).

Second, to obtain a statistically viable estimate for each parameter, we need at least 100 training examples per parameter (a good rule of thumb). The amount of training data necessary to estimate this large a number of parameters is prohibitively large. This is compounded by the fact that not all phonetic contexts occur equally frequently, which means we would need to collect orders of magnitude more data to guarantee enough examples of infrequently occurring data (or choose our data very carefully). The databases used to train systems are barely enough to provide coverage for the variability in speaker, channel, and microphone, let alone cover all possible words any speaker could speak.

This problem of insufficient training data for estimating parameters has been reduced significantly with the concept of parameter sharing across models [91,92]. The

most common method used to achieve this is the phonetic decision tree-based state-tying [91]. In this approach, a decision tree is used in an ML framework to cluster similar phonetic states. The tree is built via a set of phonetically motivated questions.

2.6.3. Parameter Initialization

A common problem with iterative algorithms such as EM is that the solution can depend on the initial values for the parameters. A simple solution has been to initialize all the models with the same set of parameters — a global mean and variance for all Gaussians and equiprobable transition probabilities. This process is often referred to as, *"flat-start"* [93]. A second approach is where the initial values for the models are computed by gathering initial means and covariance through the careful choice of exemplars for each acoustic unit being modeled [37]. The former technique has the advantage of being simple to use and it does not perform much worse than its more sophisticated counterpart. The latter technique can be very tedious and requires expert phonetic knowledge.

2.7. Summary

This chapter has reviewed a commonly used form of parameter estimation for acoustic modeling in ASR systems — Maximum Likelihood estimation of HMM parameters. The motivation for Gaussian probability distributions as models of speech variability and the use of HMMs was discussed. ML-based parameter estimation of HMM parameters in the context of Expectation-Maximization (EM) algorithm, also known as

the Baum-Welch algorithm has been described in detail. The mathematical formulation of this estimation process forms the basis for the estimation of parameters of the hybrid acoustic model formed by combining SVMs and HMMs which will be discussed in a later chapter. The following chapter will discuss alternatives to ML, which are not explicitly geared towards improving recognition performance. These discriminative acoustic modeling schemes bear resemblance to the optimizing criterion for SVMs.

CHAPTER 3

DISCRIMINATIVE TECHNIQUES FOR SPEECH RECOGNITION

The previous chapter reviewed the theoretical framework for ML-based HMM parameter estimation and some practical issues that need to addressed to make the process efficient. ML-based estimation is however tangential to the goal of classifiers in general and speech recognizers in particular. The estimation process tries to optimize the modeling ability of the acoustic models without access to a measure of their classification ability. In reality, better classification is the ultimate goal of the speech recognizer. In this chapter we look at powerful HMM parameter estimation techniques and classifiers that use some form of discriminative information while achieving better classification. This brief summary of contemporary discriminative techniques will form a backdrop for the motivation behind using Support Vector Machines to improve speech recognition.

The primary difference between maximum likelihood-based HMM parameter estimation and other discriminative techniques is that the objective criterion in the latter includes the probability of the data given that the wrong model was used [31]. Under discriminative-based estimation, the optimization process can effectively trade-off rejection of out-of-class examples while simultaneously learning to optimally represent

in-class examples. The motivation for a discriminative technique could be either based on an information theoretic concept or reduced classification error [31]. This chapter reviews techniques widely used for discriminative training of HMMs. Neural networks also fall under the class of discriminative classifiers since they learn the decision surfaces using both negative and positive examples for any particular class akin to the SVM classifiers. Neural network architectures which are widely used for speech recognition are also briefly discussed here, and compared to conventional HMM approaches.

3.1. Maximum Mutual Information (MMI)

The mutual information, I, between variables X and Y is defined as the average amount of uncertainty about the knowledge of X given knowledge of Y [94,95]. Mathematically this can be defined as:

$$I(X;Y) = H(X) - H(X/Y) \quad . \tag{31}$$

The conditional entropy of X given Y is given by

$$H(X/Y) = \sum_{x,y} P(x,y)\log P(x/y) = -E[\log P(x/y)] \quad . \tag{32}$$

Having defined mutual information, we now pose the speech recognition problem in the same framework. Let W and O denote the random variables corresponding to the words and observation vectors. The uncertainty in the word sequence given the acoustic observations is the conditional entropy of W given O,

$$H(W/O) = H(W) - I(W;O) \quad . \tag{33}$$

Note that we do not know $P(W, O)$ in general and need to estimate a parametric fit. The conditional entropy of the words given the acoustic observations can be shown to obey the following inequality:

$$H_{\lambda}(W/O) \geq H(W/O) \ , \tag{34}$$

where λ denotes a particular parametric fit to the actual distribution [19]. The equality holds only if $P_{\lambda}(W/O) = P(W/O)$. Thus by minimizing (33), we can get an estimate of the conditional distribution that minimizes the uncertainty of the data given the model. This minimization is equivalent to the maximization of the mutual information (MMI), $I(W;O)$, under the assumption that $H(W)$ is fixed.

Similar to the ML-based estimation of HMM parameters, we define an objective function, L_{MMI}, for the MMI estimation of the parameters as

$$L_{MMI}(\lambda) = I_{\lambda}(X;Y) = H_{\lambda}(W) - E[\log P_{\lambda}(w/o)] \ . \tag{35}$$

This objective function is the mutual information of the words given the acoustic observations under the parametric distribution λ. In this formulation we assume that we have acoustic data for several utterances from a training set and that we can represent each word in an utterance as a composite HMM composed of a concatenation of the HMMs representing the underlying acoustic units (phonemes, for example).

Replacing the expectations by the sample averages and assuming the training data consists of R utterances,

$$
\begin{aligned}
L_{MMI}(\lambda) &= -\frac{1}{R}\sum_{r=1}^{R}\log P_{\lambda}(w_r) \\
&\quad + \frac{1}{R}\sum_{r=1}^{R}\log\frac{P_{\lambda}(o_r|M_r)P_{\lambda}(w_r)}{P_{\lambda}(o_r)} \qquad . \qquad (36) \\
&= \frac{1}{R}\sum_{r}^{R}\{\log P_{\lambda}(o_r|M_r) - \log P_{\lambda}(o_r)\}
\end{aligned}
$$

In the above w_r is the word sequence in the r^{th} utterance with a corresponding composite model M_r. o_r are the set of observation vectors corresponding to the r^{th} utterance. The first term in the above equation is the likelihood of the data given the model. Maximizing L_{MMI} can be achieved by maximizing this likelihood, which is equivalent to ML estimation.

However L_{MMI} can also be maximized by simultaneously maximizing the first term in the RHS of (36) and minimizing the second term. The second term, the probability of the acoustic data under a particular parametrization of the model, is what differentiates MMI from ML-based estimation. The probability of the acoustic data can be defined in terms of the probability of generating all possible word sequences.

$$
P(o_r) = \sum_{s} P(o_r|M_{rs}) \cdot P(M_{rs}), \qquad (37)
$$

where s represents any possible word sequence and M_{rs} represents the composite

acoustic model for a given word sequence. Since the probability of the observation sequence includes information comprised of both the correct and the incorrect hypothesis, this optimization process is more discriminative than the traditional ML-based estimation.

3.2. Practical Issues in MMI Estimation

The steepest descent method is the simplest and most commonly used iterative technique to minimize a multivariate function [96]. At the end of each iteration, the values of the parameters are updated in the direction in which the objective function decreases the most. The change in the parameter value is a constant proportion of the gradient of the objective function with respect to the parameter. The constant proportion is commonly referred to as the learning rate.

This procedure can be concisely written as

$$x_{k+1} = x_k - \eta \nabla L_{MMI}(x_k) , \tag{38}$$

where, x_k is the value of x, a parameter of the HMMs being estimated, in the k^{th} iteration, $\Delta L_{MMI}(x_k)$ is the gradient of the objective function with respect to the model parameter and η is the learning rate. The value of the learning rate is typically computed as a line search over $L_{MMI}(x_k - \eta \nabla L_{MMI}(x_k))$. This can be an expensive process if the number of parameters being estimated is large.

Steepest descent can converge slowly depending on the slope of the optimizing surface. The direction of descent and the magnitude of the gradient play a role in defining

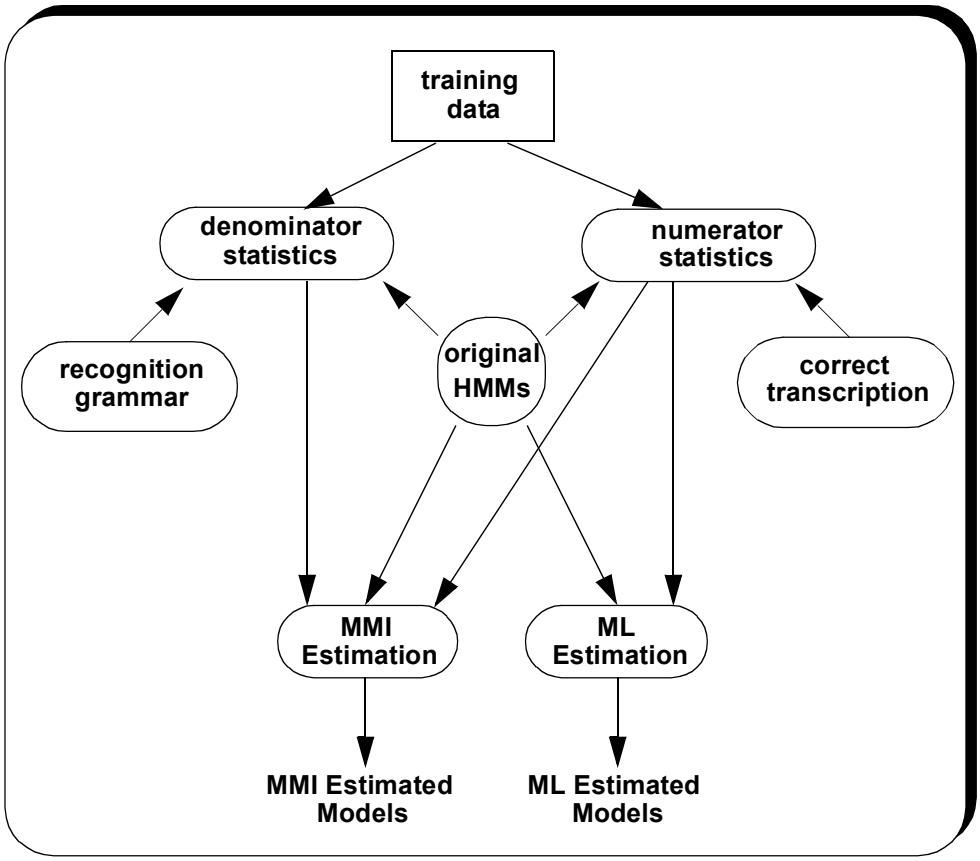

Figure 9. Practical implementation scheme for MMI estimation of HMM parameters.

the convergence properties of the optimization process. At places where the function surface is fairly flat, the gradient vector has a small magnitude, and reaching the optimum value will be slow. Similarly when the function is steep, the gradient may be large and the parameter can be updated to a value that may overshoot the optimal solution and thereby result in oscillations.

Adding momentum terms [20] is a common approach to remedy these problems. We can introduce a new term, ζ, into (36):

$$\Delta x_k = (1-\zeta)\eta \nabla L_{MMI}(x_k) + \zeta \Delta x_{k-1} \ . \tag{39}$$

This formulation uses the amount of change in the parameter value during the previous iteration in addition to the steepest descent update. In the case where consecutive iterations have the same sign for the gradient, the amount of update is large, which improves the convergence rate in flat regions of the optimization curve. When consecutive iterations have opposite signs for the gradient, the update amount is smaller, which limits the amount of oscillation about a peak in the optimization curve.

Fig. 9 demonstrates the procedure involved in estimating models using an MMI framework. Notice that the ML models are also obtained as a by-product at the end of this procedure. The box labeled “numerator statistics” includes the optimization of the first term in the RHS of (36) and “denominator statistics” contains the optimization of the second term in the RHS of (36). A recognition grammar which defines all possible word sequences (and hence all competing words for any given word) is used to guide the optimization process. In large vocabulary applications the number of competing words for a given word may be excessively large and may not be the result of acoustic confusibility. To save computations, instead of a recognition grammar, N-best lists or word-graphs [97,98] are used because the number of alternate word sequences is limited. Those word sequences retained are the result of acoustic confusions, and hence important to our goal of building classifiers with improved discrimination ability. In a later chapter we will look at a very similar procedure used to generate data to train SVM classifiers.

3.3. Minimum Classification Error (MCE)

Thus far we have reviewed parameter estimation using traditional ML and

discriminative MMI techniques. However, neither of the two explicitly attempt to optimize the primary goal of a speech recognizer — minimum classification rate. In this section, we examine a parameter estimation technique that directly minimizes errors. A minimum classification error (MCE) criterion is not limited to HMM parameter estimation and has been used to optimize several types of classifiers including learning vector quantizers and recurrent neural networks [31]. The gist of MCE optimization is that we define a loss function in terms of the trainable parameters of the classifier that is proportional to the classification error. This loss function is then minimized using a suitable gradient-based technique [99]. MCE training does not necessarily involve the estimation of probability distributions and hence no underlying probability distribution needs to be assumed [20]. This circumvents a major drawback of ML estimation.

3.3.1. Generalized Probabilistic Descent

MCE is a powerful technique because it allows us to build classifiers that perform close to the Bayes error rate [41]. However, the technique would not have been successful if not for the existence of an efficient method for this optimization. Generalized Probabilistic Descent (GPD), which is based on Amari's Probabilistic Descent theorem, forms the basis of the optimization procedure used in MCE [100,101]. This theorem states that for an infinite sequence of random samples o_t and step size sequence c_t that satisfies the conditions,

1. $$\sum_{t=1}^{\infty} c_t \to \infty, \text{ and} \tag{40}$$

2. $$\sum_{t=1}^{\infty} c_t^2 < \infty \tag{41}$$

adapting the system parameters according to

$$\Lambda_{t+1} = \Lambda_t - c_t U \nabla l_k(o_t, \Lambda_t), \tag{42}$$

converges with a probability of one to a local minimum of the overall loss, $L(\Lambda)$. U in the above equation is a positive definite matrix that specifies separate update rates for each estimated parameter.

The key to this approach is that the overall loss is never computed. Instead, we optimize based on the local loss functions, $l_k(\boldsymbol{o}, \Lambda)$. For the purpose of MCE, we specify the local loss to be a function of the classification error. However, the GPD framework can be used for any other form of local loss functions. Though in theory, an infinite number of samples are required for convergence, in practice it has been found the algorithm will converge using a finite amount of data.

3.3.2. MCE Theory

The misclassification error measure in a classification problem can be defined in terms of discriminant functions of the k classes, C_k. In typical speech recognition

applications these classes are either words in the application vocabulary or phonemes used as models internal to the system. We can choose a misclassification error such that it takes a value of zero for all correct classifications and non-zero values for misclassifications. This measure is not extremely useful because it does not provide a degree of separation between the correct and incorrect classes. In practice a measure with a gradual slope is preferred. One such popular misclassification error measure [31] is

$$d_k(\boldsymbol{x}, \Lambda) = -g_k(\boldsymbol{x}, \Lambda) + \left[\frac{1}{L-1} \sum_{j \neq k} g_j(\boldsymbol{x}, \Lambda)^{\psi} \right]^{\frac{1}{\psi}}, \tag{43}$$

where g_k is the discriminant function corresponding to the k^{th} class, ψ controls the contribution of each misclassification towards the error metric and L is the number of classes in the classification problem [31]. When ψ is large, the most confusable class contributes the most to the summation.

This behavior is consistent with an N-best list processing paradigm or Viterbi approximation commonly used in many speech recognition systems [102,103]. In a Viterbi approximation, the summation over the discriminant functions is replaced by the maximum discriminant. Similarly, for an N-best list processing, summation over the discriminants is replaced with the summation over the highest N conditionals.

3.3.3. Practical Issues in MCE

Thus far we have introduced the general concept of MCE and its implementation

using GPD. We now focus on using the MCE framework in HMM parameter estimation. We start with the definition of the discriminant function in terms of the parameters of an HMM. We have several choices for the form of the discriminant function. The primary requirement is that the discriminant function can be used as a distance metric to compare classes. For this reason, the likelihood of the class, C_j, in terms of the transition and observation probabilities is often used.

The likelihood is computed as the probability of all possible state sequences, Θ^p, for the given data. A closed form expression for one particular state sequence can be written as

$$f(\boldsymbol{x}_1^T, \theta^p \,|\, \Lambda) = \prod_{t=1}^{T} a_{\theta_{t-1}^p \theta_t^p} \cdot b_{\theta_t^p}(\boldsymbol{x}_t) \,, \tag{44}$$

where a and b are the HMM transition and observation probabilities, respectively. Using the above definition of the likelihood, the discriminant function for the j^{th} class can be defined as

$$g_j(\boldsymbol{x}_1^T, \Lambda) = \log\left[\sum_p [f(\boldsymbol{x}_1^T, \theta^p \,|\, \Lambda)]^{\xi}\right]^{\frac{1}{\xi}}. \tag{45}$$

Note that when ξ is large, the most probable state sequence dominates the summation and we approach a Viterbi solution.

A loss function, d, can now be defined as the misclassification error measure. Equation (43) defines a commonly used loss function. This loss function is then minimized using gradient descent approaches similar to MMI estimation.

From Amari's theorem, convergence to the local MCE optimum involves optimizing local loss functions [101]. In general there are certain desirable properties for loss functions since GPD involves gradient computations. Near-binary functions are a desirable form for loss functions. Loss functions need to be first-order differentiable to apply GPD. A commonly used loss function that satisfies the above requirements is the sigmoid function:

$$l(d) = \frac{1}{1 + e^{-\alpha d}}, \tag{46}$$

where d is the misclassification error measure.

3.3.4. Relationship of MCE to Bayes Decision Theory

The primary difference between MCE and other HMM parameter optimization techniques like ML or MMI can be better illustrated by noting the link between MCE and the Bayes decision theory. In an M class problem, the probability of error given by Bayes rule is,

$$P_{error} = \sum_{k=1}^{M} \int_{X_k} P(\boldsymbol{x}, C_k) I(\boldsymbol{x} \in C_k) d\boldsymbol{x}, \tag{47}$$

X_k represents the set of misclassified data samples and can be defined as,

$$X_k = \{ \boldsymbol{x} \in \mathrm{X} | P(C_k|\boldsymbol{x}) \neq \max_i P(C_i|\boldsymbol{x}) \} \quad (48)$$

P_{error} is the lower bound for the error in this classification task since it requires an exact knowledge of the underlying probability distribution. This error measure is an inherent property of the classification problem and does not include errors accrued due to modeling assumptions. In reality, P needs to be estimated from data which is at best an approximation to the actual distribution. Using a similar formulation, we can compute the error conditioned on the discriminant functions $g_k(\boldsymbol{x}, \Lambda)$ as,

$$P_{\Lambda, error} = \sum_{k=1}^{M} \int_{X_k(\Lambda)} P_\Lambda(\boldsymbol{x}, C_k) I(\boldsymbol{x} \in C_k) d\boldsymbol{x} , \quad (49)$$

and

$$X_k(\Lambda) = \{ \boldsymbol{x} \in \mathrm{X} | g_k(\boldsymbol{x}, \Lambda) \neq \max_i g_i(\boldsymbol{x}, \Lambda) \} \quad (50)$$

P_Λ is defined over the region in the observation space that is determined by the choice of the classifier, its parameters and the decision rule. This is different from P which is defined over the region in the observation space determined by applying the Bayes rule assuming knowledge of the true posterior probabilities.

By defining appropriate loss functions, P_Λ can be made to approach the Bayes error. In fact, when

$$\forall k, \left(g_k(\boldsymbol{x}, \Lambda) \neq \max_i g_i(\boldsymbol{x}, \Lambda) \right) \equiv I\left(P\langle C_k|\boldsymbol{x}\rangle \neq \max_i P\langle C_i|\boldsymbol{x}\rangle \right), \tag{51}$$

$$P_{\Lambda,\, error} = P_{error}. \tag{52}$$

where I is an indicator function. Achieving the Bayes minimum error does not necessarily imply that the discriminant functions represent the posteriors. It only implies that the class with the greatest discriminant function corresponds to the class with the greatest *a posteriori* probability. What this means is that, in order to obtain minimum classification error through the application of the Bayes rule, we need not explicitly model the posterior probabilities. This is by the far the biggest difference between the principle of MCE and ML or MMI. When MCE is used, a wrong choice for the form of the probability density functions in an HMM does not hurt performance as much as it does when ML-based optimization is used. As described in the next chapter, SVMs are also optimized via the estimation of discriminant functions.

3.4. Neural Network-Based Approaches

Though neural networks have been studied for several decades now, it was only since 1986, with the introduction of the back-propagation training algorithm, that they have found widespread use in several complex classification problems [45-47]. The back-propagation algorithm is a simple scheme wherein the gradient of the error between the network output and the expected output is propagated back from the output layer to the input layer while modifying the weights along the connections [104-108]. The weights are

modified so as to satisfy some optimization criterion — minimum mean squared error is commonly used.

It has been shown that networks with sufficient number of layers and nodes can model complex non-linear decision regions and functions [104]. The estimation of the parameters of the neural networks can be considered discriminative, in that the weights of the connections are updated based on training examples belonging to all the classes in the task, unlike ML-based techniques. Also, unlike ML-based techniques, training for all the classes is done simultaneously using all the data instead of training classifiers using in-class data alone. Though the MLP has been successful on several classification tasks, modifications to the network are required to handle certain characteristics of speech data — most notably the dynamic nature of the evolution of speech

In an HMM, the likelihood of generating an observation is dependent only upon the state and is independent of all other observations. This is not a valid assumption since the speech signal continuously evolves during the observation period (albeit slowly compared to the interval), and this evolution makes consecutive observations dependent on one another. The fact that the human speech production system is a complex dynamical system incapable of sudden transitions makes this assumption significant. The following modifications to the basic MLP neural network aim at harnessing the modeling power of neural networks while reducing the effect of the observation independence assumption.

3.4.1. Time-delay Neural Networks

The time delay neural network (TDNN) architecture was developed initially for phoneme recognition [104,105]. In this constrained form of an MLP, connections are time-delayed (i.e. there is no instantaneous propagation of excitation from the input layer

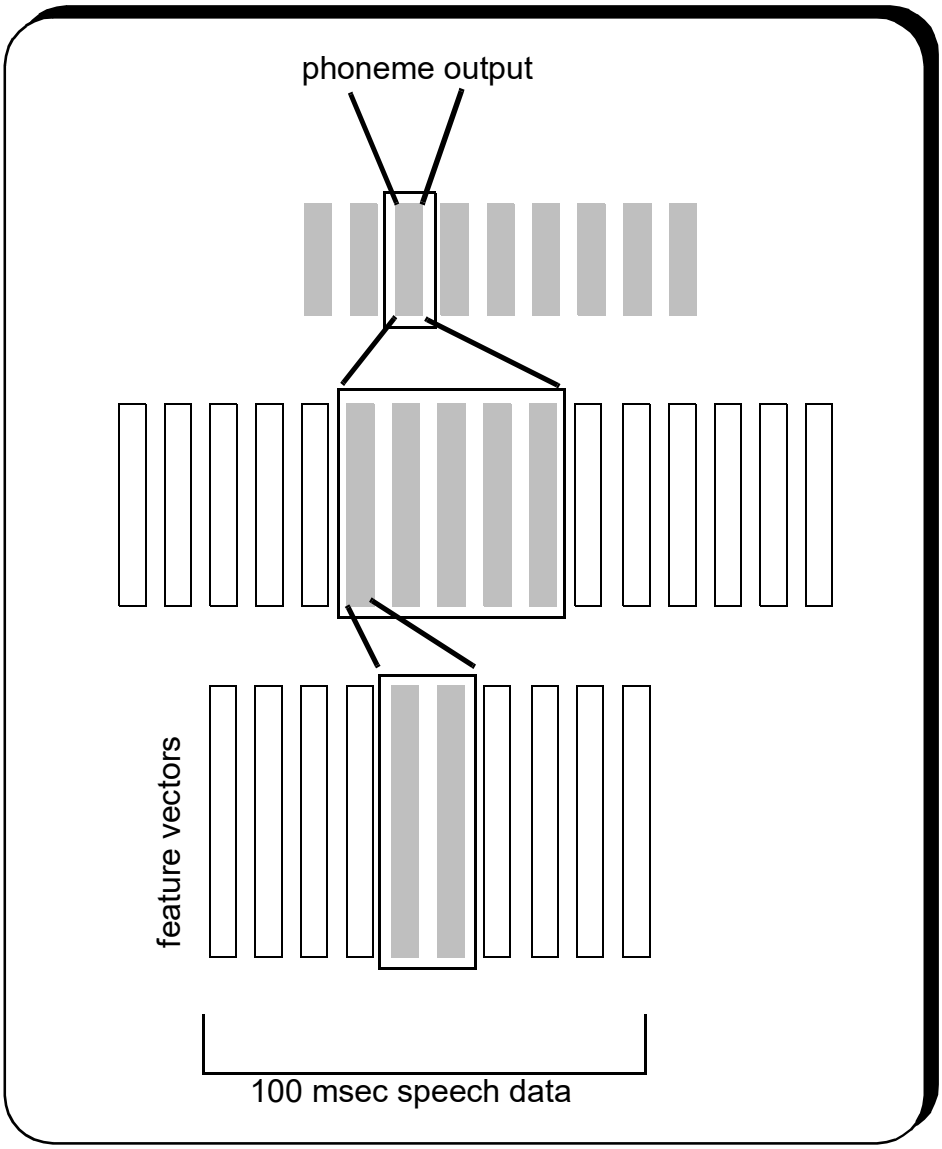

Figure 10. An example of a time delay neural network used for phone recognition.

to the output layer) to cope with varying segmentations of the incoming speech data. The idea is to empower the neural network with the ability to match segments irrespective of their occurrence in the input data stream. Fig. 10 shows a simple TDNN implementation. In this example, the input layer of the network is exposed to 10 frames, or 100 msec., of speech data. The first hidden layer groups two frames every 10 msec. and passes these abstract features to the next hidden layer which groups five outputs of the first hidden layer every 10 msec. This second hidden layer is connected to the output nodes that predict the phoneme class. A total of 10 frames of input labeled speech data are used to predict this phone label, as shown in Fig. 10.

Some of the special characteristics of the TDNN include:

- the time delays are hierarchically structured so that the units closer to the output layer can integrate information from a wider temporal context;
- the connections are shared along the time axis to reduce the number of trainable parameters in an attempt to improve generalization.

The manner in which the network connections are made forces the network to learn speech events independent of the exact location of the events in the input. This makes the network shift-tolerant or shift-invariant.

3.4.2. Recurrent Neural Networks

Though TDNNs have been successful in many applications, their use in speech recognition has been limited to small tasks. One of the main drawbacks of TDNNs is that the temporal context that is useful for the classifiers is predetermined. The network itself cannot find an optimal time window to learn contextual information. Another drawback

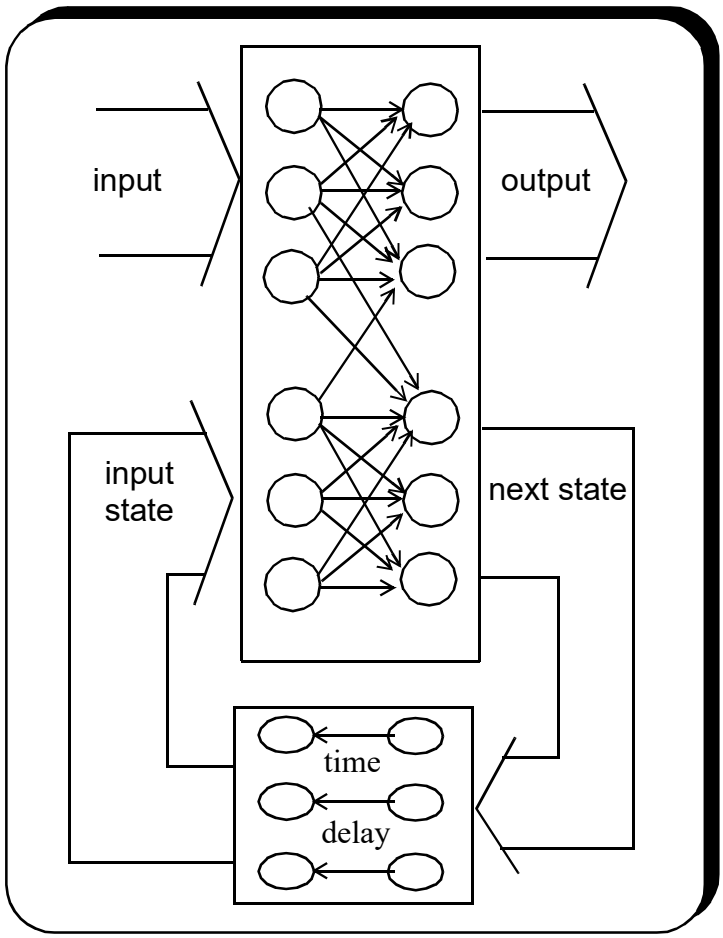

Figure 11. A simple recurrent neural network.

with TDNNs is the need to present the network with multiple frames of segmented data at a time, unlike HMM based systems where data is provided to the classifiers one frame at a time and the segmentation is learned from the data.

This problem can be circumvented by adding feedback to an MLP. Feedback provides an efficient method to add context to the neural network without having to feed multiple frames of data to the classifiers. Feedback also saves resources in terms of the size of the networks. Recurrent neural networks (RNN) are an example of neural networks with feedback [106,107]. RNNs have an advantage of learning contextual effects in a

data-driven fashion. RNNs also have the ability to learn long-term contextual effects. Traditionally, contextual effects have been dealt with in HMM-based systems by using explicit context-dependent models or by increasing the dimensionality of the feature vectors to include gradients of the features [16,55,57].

Fig. 11 shows a simple RNN structure. The current input and current state form the inputs to the network. The two inputs are fed forward through the network to generate an output vector and the next state vector. The next state vector is fed back as input to the RNN via time delay unit. The parameters of an RNN are estimated using an efficient estimation procedure called back-propagation through time (BPTT) [106]. The basic idea behind BPTT is that an RNN can be unfolded in time to represent an MLP where the number of hidden layers is equal to the number of frames in the input sequence. Training an RNN can now be executed in a similar way as the standard MLP using back propagation with the restriction that the weights at each layer be tied [104]. Since RNNs process one frame at a time, they have been more popular in hybrid connectionist systems where the neural networks are used as posterior probability estimators and are embedded in an HMM framework [109,111]. This form of a hybrid architecture is similar to the hybrid system developed as part of this dissertation [73,75,76,77]. A detailed analysis of the hybrid architectures used in connectionist systems will be discussed in a later chapter.

3.4.3. Minimum Mean Square Error and Bayes Decision Theory

We have thus far seen two common neural network architectures used in speech recognition systems. Minimizing the mean-squared error is commonly used as the

optimizing criterion for estimating the weights of these networks. The following derivation shows why mean squared error is a good choice for optimizing the weights of a network in a typical classification scenario.

Consider a two class classification problem where y is equal to 1 for samples in class C_1 and zero otherwise. We then have,

$$E[y|\mathrm{x}] \;=\; P(y=1|\mathrm{x}) \;=\; P(C_1|\mathrm{x}). \tag{53}$$

The above equation indicates that the expected value of the output of the MLP is nothing but the class posterior probability. If the goal of the optimization criterion is to minimize the mean-squared error between the target output (binary in this case) and the MLPs response, the optimization criterion in terms of the MLPs output f can be written as,

$$E[(y-f(x))^2|x] \;=\; E[((y-E[y|x])+(E[y|x]-f(x)))^2|x] \tag{54}$$

$$\begin{aligned} &= E[(y-E[y|x])^2|x] + E[(E[y|x]-f(x))^2|x] \\ &\quad + 2E[(y-E[y|x])|x](E[y|x]-f(x)) \end{aligned} \tag{55}$$

$$= E[(y-E[y|x])^2|x] + (E[y|x]-f(x))^2 \tag{56}$$

$$\geq E[(y-E[y|x])^2|x] \tag{57}$$

Thus, when mean squared error is minimized, the output of the MLP, $f(x)$, is equal to the posterior probability, $P(C_1|\mathrm{x})$ based on (53). In an ideal situation if an MLP has enough parameters to model the posterior probability, it can implement a Bayes decision rule

accurately using the minimum mean-squared error as the optimizing criterion. However, modeling the posterior probability density is not always easy even with a significant number of parameters [31]. In spite of the above limitation of MLPs, they are theoretically better classifiers than Gaussian classifiers because of the discriminative framework in which the parameters are estimated. When large amounts of data are available, complex MLPs can model posterior probabilities very well. The classifiers that result typically model complex non-linear decision regions similar to non-linear kernel-based SVMs.

3.5. Summary

This chapter has introduced common discriminative approaches to acoustic modeling in speech recognition. In HMM-based systems, the discrimination ability of the Gaussians is improved by using optimizing criteria like MMI and MCE which include both in-class and out-of-class data in the estimation process. MCE is especially elegant in that it uses the fact that in order to achieve good classification, the estimation of the posterior probabilities is not as important as it appears. ML and MMI suffer from the fact that both the estimation procedures expend effort in modeling posteriors while not guaranteeing improved classification performance. MMI is more discriminative that ML in that out-of-class data is also involved in the optimization process. This allows for simultaneously learning a good representation for in-class data while discriminating out-of-class data.

An alternative to HMM-based discriminative techniques are neural network-based systems. It has been shown that estimating the parameters of the network via the minimum

mean-squared error criterion implies that the output of the network is in fact a good estimate of the posterior probability. This posterior probability is the key to implementing a Bayes decision rule. Two commonly used neural network architectures for speech recognition, time delay neural network (TDNN) and recurrent neural network (RNN), were discussed. Though TDNN has found limited use in continuous speech recognition, the RNN architecture has been used in several successful connectionist hybrid speech recognition systems.

CHAPTER 4

SUPPORT VECTOR MACHINES

We have seen in the previous chapter the motivation for the use of discriminative parameter estimation algorithms for speech recognition. However, with HMMs, we still aim to model speech by estimating a representation of the speech sounds. At a high level, speech recognition can be viewed as a classification problem. In that respect one would expect better performance with classifiers that estimate decision surfaces directly rather than those that estimate a probability distribution across the training data.

A Support Vector Machine (SVM) is one such machine learning technique that learns the decision surface through a process of discrimination and has good generalization characteristics [68]. SVMs have been proven to be successful classifiers on several classical pattern recognition problems [25]. In this chapter we take an in-depth look at the mathematical foundation of SVMs and implementation issues that come about when we apply these to complex classification tasks like speech recognition.

4.1. Risk Minimization

Suppose that the training data consists of pairs,

$(x_1, y_1), (x_2, y_2), \ldots$

where the x 's are the input observations and y 's are the output observations. The goal of a learning machine is to learn the mapping $y = f(x)$. We assume that the training data has been drawn randomly and independently based on the joint distribution $P(x, y)$. To learn the unknown mapping, we can either estimate a function that is "close" to the joint distribution under an appropriate metric or learn an optimal predictor of the system's output. In the former case, it is not sufficient for us to estimate a good predictor of the output. The goal is to estimate P , the joint distribution. However, for the purposes of data classification, we pursue the latter approach where the goal is to learn an optimal predictor.

The learning process is therefore a process of choosing a function from a set of functions defined by the construction of the learning machine. For example, in a neural network classifier [48], the problem reduces to that of finding the weights of the connections in a predefined network. Since the network structure has been defined a priori, the set of networks from which the optimal network needs to be chosen is a finite set. The optimal network is chosen based on some optimality criterion that measures the quality of the learning machine.

Let us define a term, *risk,* which measures the quality of the chosen function. The operating space is a subset Z of an n-dimensional vector space R^n , which is the union of the input vector space and the output space. Let us also assume the existence of a set of functions $\{g(z)\}, z \in Z$ and a functional R , that measures the risk as,

$$R(g(z)) = \int L(z, g(z)) dP(z), \tag{58}$$

where L is an appropriately defined loss function and P is the previously defined joint probability distribution. The problem of *risk minimization* can then be defined as one that minimizes the functional given by (58) for a specific training data set. In reality the minimization process involves finding the optimal parametrization for the function $g(z)$ which can be parametrically represented as $g(z, \alpha), \alpha \in \Lambda$. The minimization involves finding the best parametrization α^* such that $g(z, \alpha^*)$ is the optimal function from the set of functions $\{g(z, \alpha)\}$. The above parametrization does not necessarily imply that the problem is restricted to parametric forms since α can be a scalar, a vector or any other abstract functional element.

With the above modifications to the definition of the minimization problem, the risk can be rewritten as,

$$R(\alpha) = \int Q(z, \alpha) dP(z), \qquad \alpha \in \Lambda, \tag{59}$$

where,

$$Q(z, \alpha) = L(z, g(z, \alpha)). \tag{60}$$

The function Q is now called the *loss function*. Choosing an appropriate value for α to minimize the functional defined in (59) is called *risk minimization*. Minimizing the risk functional is not a trivial problem because the form or the parametrization of the joint distribution $P(z)$ is not known a priori.

The problem can be simplified significantly if we minimize a variation of the risk defined previously. For example, instead of minimizing the risk defined in (59), we can minimize

the measured mean risk defined as,

$$R_{emp}(\alpha) = \frac{1}{l}\sum Q(z_i, \alpha), \qquad \alpha \in \Lambda, \tag{61}$$

which is called the *empirical risk.* In the above formulation we assume that we have access to l training observations $z_1, z_2, \ldots z_l$. R_{emp} is therefore the mean error computed from the fixed number of training samples assuming that the training samples are uniformly distributed. Minimization of the above functional is called *Empirical Risk Minimization (ERM)* and is one of the most commonly used optimization procedures in machine learning. ERM is computationally simpler than attempting to minimize the actual risk as defined in (59). ERM circumvents the need for the estimation of the joint probability density function P. In many cases ERM provides a good quality learning machine. A variety of loss functions can be used for the optimization process. One such example is,

$$Q(x, y) = |y - f(x, \alpha)| \ , \tag{62}$$

where y is the output of the classifier and x is the input vector. This form of a loss function is common in learning binary classifiers. For example, to estimate the parameters of a multi-layered perceptron [53] using the back-propagation algorithm, a loss function representing the squared error is used.

The issue of the quality of the learning machine is not addressed in its entirety when we talk about ERM. There could be several configurations of the learning machine

which give us the same empirical risk (which is a zero in the case of binary classifiers). How does one choose the best configuration? To better answer this question we need to analyze the relationship between the actual risk and the empirical risk.

Suppose that the minimum empirical risk is obtained using the function $Q(z, \alpha_l)$, where the subscript l is equal to the size of the training sample. Let the minimum actual risk be obtained using the function $Q(z, \alpha_0)$. There are two issues that need to be addressed. First, we need to know the risk achieved using $Q(z, \alpha_l)$. Second, we need to know how close this risk is to the risk obtained using $Q(z, \alpha_0)$.

Vapnik [28] proved that bounds exist for the actual risk such that,

$$R(\alpha) \leq R_{emp}(\alpha) + f(h) \ . \tag{63}$$

The quantity h is the Vapnik-Chervonenkis (VC) dimension [28,68] and is a measure of the capacity of a learning machine. More formally, it is defined as the largest dimension of vectors that can be shattered by the functions $Q(z, \alpha_l)$. We say that a sample X of dimension m is *shattered* by F, or that F *shatters* $\boldsymbol{X}$, if F gives all possible classifications of $\boldsymbol{X}$. Inequality (63) addresses the generalization ability of the learning machine. For example, if $f(h)$ is small, the machine generalizes well because the actual risk is guaranteed to be close to the empirical risk which has been already minimized via the principle of ERM.

To better qualify the above statement, suppose that the empirical risk obtained using the old data set is zero. This fixes R_{emp} and the actual risk is now bounded by $f(h)$.

Suppose we now receive a new set of data that does not include any of the l examples used previously. For a machine that generalizes well, we should be able to predict with a high degree of confidence that the empirical risk obtained using this new data, R_{emp}^*, will also be close to zero. However, from (63), we note that the actual risk over the data can be as high as $f(h)$. Therefore when $f(h)$ is large, R_{emp}^* can be as high as $f(h)$ and R_{emp} cannot be used to effectively predict the performance of the machine on an unseen dataset. In other words, the machine fails to generalize well when $f(h)$ is large and can be an ineffective classifier for unseen datasets.

For loss functions in the form of indicator functions — which is true in the case of binary classifiers, the second term in the r.h.s. of (63) is,

$$\frac{\varepsilon(l)}{2}\left(1+\sqrt{1+\frac{4R_{emp}(\alpha_l)}{\varepsilon(l)}}\right) \tag{64}$$

where α_l is the parameter set that defines the learning machine and $\varepsilon(l)$ is the measure of the difference between the expected and empirical risk [68]. The error term, $\varepsilon(l)$, can be written in terms of the VC dimension and the size of the training set as,

$$\varepsilon(l) = 4\frac{h(\log(2l/h+1)) - \log\eta/4}{l}, \tag{65}$$

where h is the VC dimension [68].

Equation (65) provides us with a good method for comparing system configurations optimized using empirical risk minimization. When l/h is large, ε and

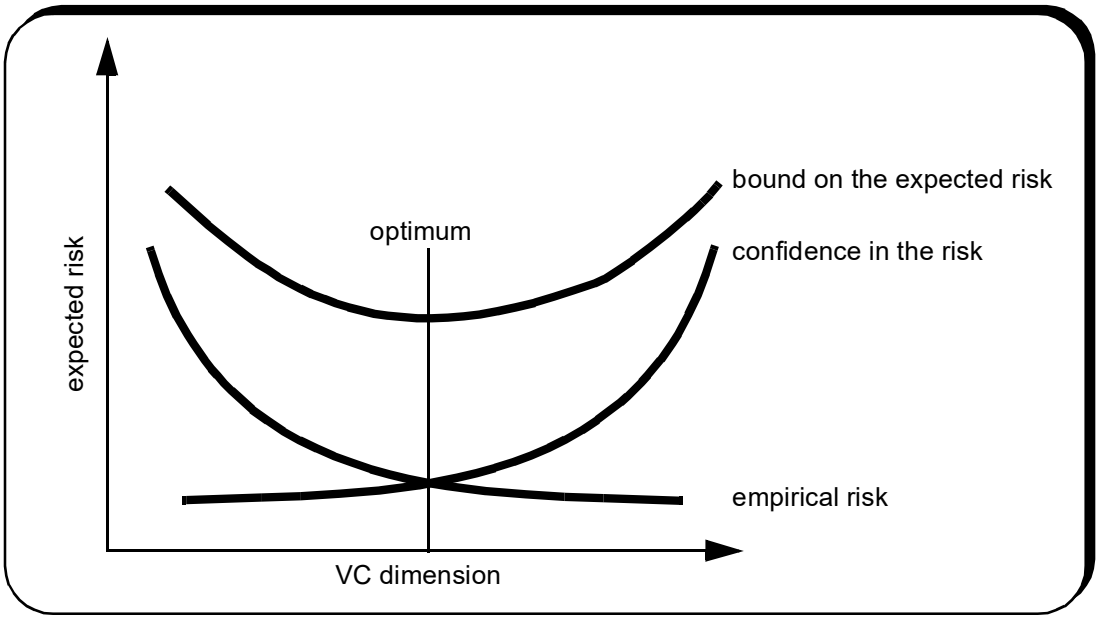

Figure 12. The optimal classifier needs to achieve the lowest bound on the risk.

$f(h)$ are both small. This implies that the expected risk tends towards the empirical risk. With this, we can guarantee both a small empirical risk (training error) and good generalization — an ideal situation for a learning machine. On the other hand, when l/h is small, both ε and $f(h)$ are large. Under this condition, a small empirical risk does not guarantee a small expected risk. Thus, the system is not guaranteed to generalize well. In this case, both terms on the r.h.s. of (63) need to be minimized simultaneously. From (65), we see that the error term, ε, monotonically increases with the VC dimension. This implies that the confidence in the empirical risk decreases monotonically with the VC dimension as does the generalization ability. These observations are depicted in Fig. 12.

The principle of structural risk minimization (SRM) is an attempt to identify the

optimal point on the curve describing the bound on the expected risk. "The SRM principle defines a trade-off between the quality of the approximation of the given data and the complexity of the approximating function." [28]. From the above discussion, making the VC dimension a controlling variable for the generalization ability of the learning machine seems like a natural choice. In practice the principle of SRM can be implemented in two distinct flavors:

- for a fixed confidence interval optimize the empirical risk
- for a fixed empirical risk optimize (or minimize) the confidence interval

The decision to use any one of the above schemes is problem/classifier dependent.

Neural network learning uses the former procedure by first fixing the network structure and then minimizing the empirical risk using gradient descent. SVMs implement SRM using the latter approach where the empirical risk is fixed at a minimum (typically zero for separable data sets) and the SVM learning process optimizes for a minimum confidence interval. In other words, SRM is an extension of ERM with the additional constraint that a *structure* be added to the space containing the optimal function. For example, structure can be imposed on the problem of function estimation using neural networks by associating the number of hidden units or their connections to each subset. In the case of optimal hyperplane classifiers, which will be discussed in the next section, structure is imposed by the width of the margin of the separating hyperplane.

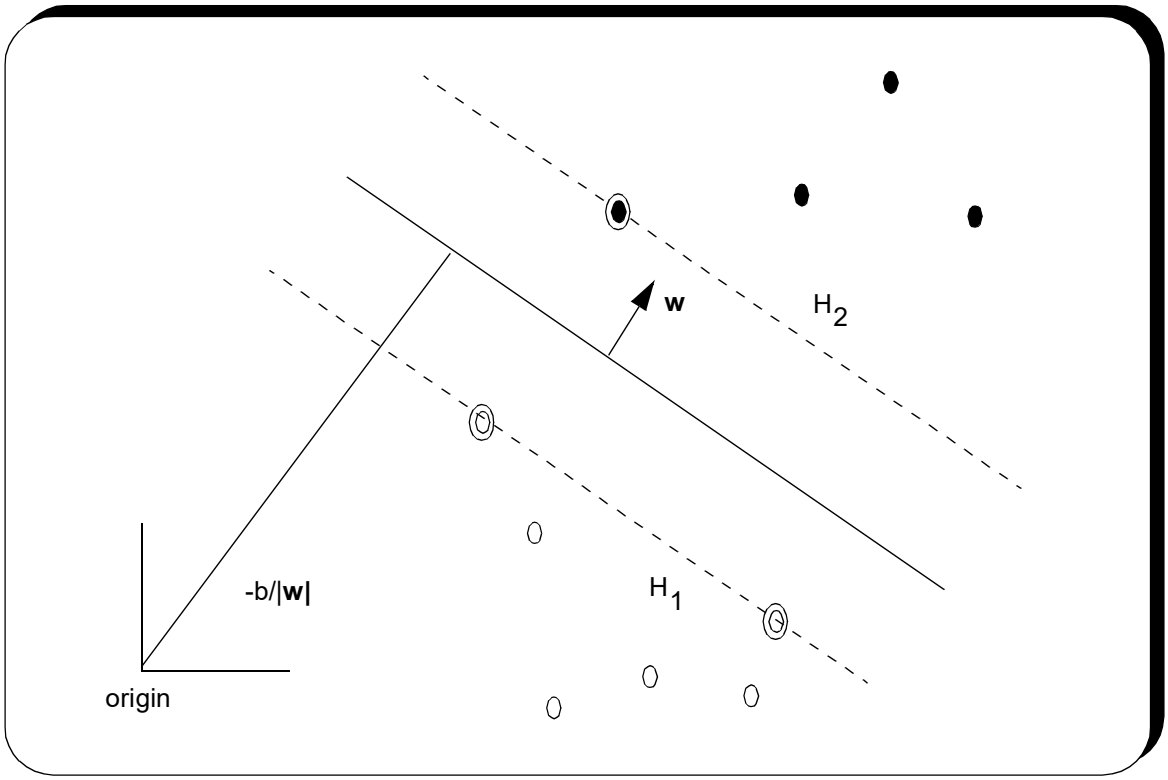

Figure 13. Definition of a linear hyperplane classifier. SVMs are constructed by maximizing the margin.

4.2. Optimal Linear Hyperplane Classifiers

The following formulation is based on the fact that among all hyperplanes separating the data, there exists a unique hyperplane that maximizes the margin of separation between the classes [28]. Fig. 13, shows a typical 2-class classification example where the examples are perfectly separable using a linear decision region. H_1 and H_2 define two hyperplanes the distance between which is called the *margin*. The closest in-class and out-of-class examples lie on these two hyperplanes. As noted earlier, the SRM principle imposes structure to the optimization process by ordering the hyperplanes based on this margin. The optimal hyperplane is the one that maximizes the margin while

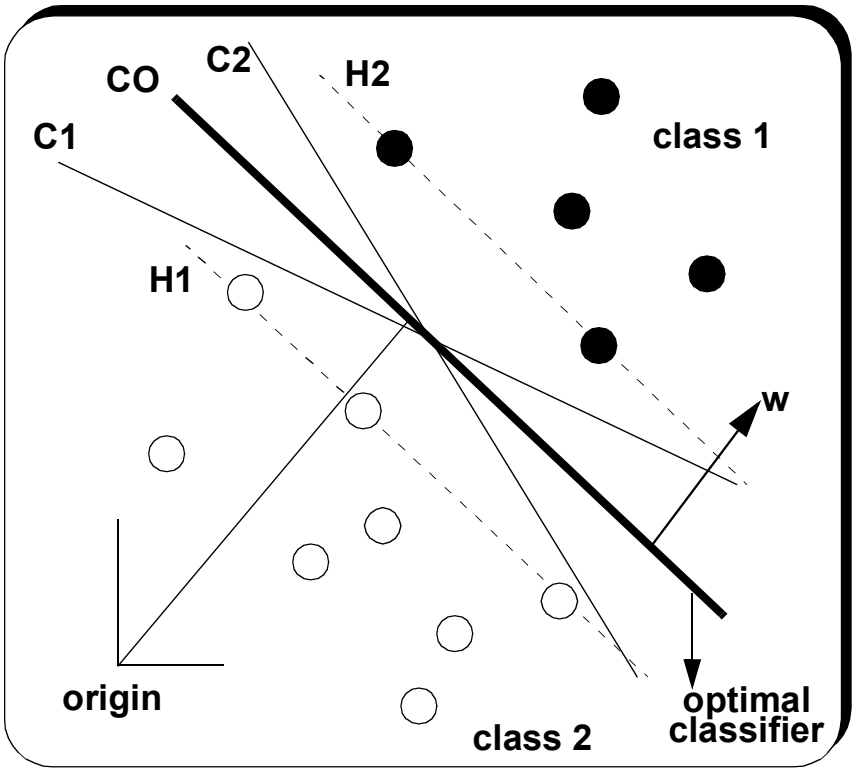

Figure 14. This is an illustration of the difference between the classifiers that result from optimization based on empirical risk minimization and structural risk minimization. Hyperplanes C0, C1 and C2 achieve perfect classification and, hence, zero empirical risk. However, C0 is the optimal hyperplane because it maximizes the margin, the distance between the hyperplanes H1 and H2. A maximal margin indirectly results in better generalization.

minimizing the empirical risk. Fig. 14 illustrates the difference between using ERM and SRM to estimate a simple hyperplane classifier. Using SRM results in the optimal hyperplane classifier.

Let $\boldsymbol{w}$ be the normal to the decision region. Let the l training examples be represented as the tuples $\{\boldsymbol{x}_i, y_i\}$, $i = 1, \ldots, l$ where $y = \pm 1$. The points that lie on the hyperplane separating the data satisfy

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0 \tag{66}$$

where b is the distance of the hyperplane from the origin. Let the "margin" of the SVM be defined as the distance from the separating hyperplane to the closest positive and negative examples. The SVM training paradigm finds the separating hyperplane which gives the maximum margin. Once the hyperplane is obtained, all the training examples satisfy the following inequalities.

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \geq +1 \qquad \text{for } y_i = +1 \tag{67}$$

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \leq -1 \qquad \text{for } y_i = -1 \cdot \tag{68}$$

The above equations can be compactly represented as a single inequality,

$$y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 \geq 0 \qquad \forall i. \tag{69}$$

Looking at the above equations with respect to Fig. 13, we see that all points satisfying the equality condition of (67) lie along the hyperplane H_1. Similarly, all points satisfying the equality condition of (68) lie along the hyperplane H_2. The distance between H_1 and H_2, also called the margin, is therefore two units. Since the normal to the hyperplane is not constrained to be of unit-norm, we need to normalize this margin by the norm of the normal vector to the hyperplane, $\boldsymbol{w}$. Therefore the margin as defined by the hyperplane is $2/\|\boldsymbol{w}\|$. For a completely separable data set, no points fall between H_1 and H_2. To maximize the margin we need to therefore maximize $1/\|\boldsymbol{w}\|$. Elegant techniques exist to optimize convex functions with constraints [88]. We therefore minimize $\|\boldsymbol{w}\|^2$, a convex function, instead. The training points for which the equality in (69) holds are called

support vectors. In Fig. 13, they are shown as data points with concentric circles that lie on either of the hyperplanes H_1 or H_2.

The theory of Lagrange multipliers [112] can be used to solve optimization problems involving convex functionals with constraints. The functional for the optimization problem in this discussion, called the Lagrangian, can be written as,

$$L_P = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \alpha_i y_i (\boldsymbol{x}_i \cdot \boldsymbol{w} + b) + \sum_{i=1}^{N} \alpha_i . \tag{70}$$

In the above equation, L_P is called the functional. The first term on the RHS, defined as half the square of the norm, is called the objective function and the other two terms are the optimization constraints. Optimization of L_P is clearly a convex quadratic programming problem since the objective function itself is convex. The above is called the *primal* formulation of the optimization problem. Since we are minimizing L_P, its gradient with respect to $\boldsymbol{w}$ and b should be equal to zero. This gives the system of equations,

$$\boldsymbol{w} = \sum_i \alpha_i y_i x_i , \text{ and} \tag{71}$$

$$\sum_i \alpha_i y_i = 0 . \tag{72}$$

We can define a *dual* problem of the same type as the primal such that the

Lagrange multipliers of the primal can be obtained as part of the solution for the dual and vice versa. In several cases optimizing the dual is easier because of the structure of the problem [88,113,114]. For the problem at hand, to arrive at the dual formulation we need to substitute (71) and (72) back into (70). Substituting for $\boldsymbol{w}$ into (70) gives,

$$L_D = \begin{aligned} &\frac{1}{2}\sum_i \alpha_i y_i \boldsymbol{x}_i \cdot \sum_j \alpha_j y_j \boldsymbol{x}_j \\ &-\sum_i \alpha_i y_i \left(\boldsymbol{x}_i \cdot \sum_j \alpha_j y_j \boldsymbol{x}_j + b \right) + \sum_i \alpha_i \end{aligned} \tag{73}$$

Using (72) in the above equation results in the final dual formulation,

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \ . \tag{74}$$

SVM learning can thus be treated as the problem of maximizing L_D with respect to α_i subject to their positivity and the constraints in (72). The positivity constraint is a direct result of the Kuhn-Tucker theorem [113] that applies to the optimization of a convex function constrained by concave functions. The theorem guarantees the existence of Lagrange multipliers that are non-negative. The next section states the theorem uses a simple example to show the use of conditions as proposed by the theorem.

Equations (66) and (71) imply that the decision function can be defined as,

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i \cdot \boldsymbol{x} + b \tag{75}$$

where the sign of f can be used to classify examples as either in-class or out-of-class. In other words the above equation defines the SVM classifier. This definition of the classifier is worth a closer look. Note that the classifier is defined in terms of the training examples. However all training examples do not contribute to the definition of the classifier. The training examples with non-zero multipliers, the Support Vectors, alone define the classifier. In other words for the purpose of classification the dataset defines how complex the classifier needs to be. For simple classification problems the number of support vectors is small and vice versa. It is also interesting to see how the complexity of the classifier scales with the number of support vectors. Since there are M dot products involved in the definition of the classifier, where M is the number of support vectors, the classification task scales linearly with the number of support vectors.

We have thus far posed the problem of hyperplane classifiers as one of optimization. However, we have not addressed the issue of the existence of such an optimum. Even if an optimal point exists, how do we guarantee that there is a single optimal point? In order to answer these questions we look at the Karush-Kuhn-Tucker (KKT) theorem that guarantees the existence of a solution and also prescribes a set of necessary and sufficient conditions. The KKT theorem has found widespread use in optimization problems specifically dealing with convex objective functions. In solving the problem of finding the optimal hyperplane we use the KKT conditions in formulating the constraints. The positivity constraint on the Lagrange multipliers as mentioned earlier is one such example. The KKT theorem is defined in Fig. 15. In the definition of the KKT theorem, ∇ ,

Let $f, \boldsymbol{h}, \boldsymbol{g}$ be three functions. Let x^* be a data point and a local minimizer for the problem of minimizing f subject to $\boldsymbol{h}(x) = 0$, $\boldsymbol{g}(x) \leq 0$. Then there exist $\lambda^* : \in \Re^m$ and $\mu^* : \in \Re^p$ such that:

$$\mu^* \geq 0 \tag{76}$$

$$\nabla f(x^*) + \lambda^{*T} \nabla \boldsymbol{h}(\boldsymbol{x}^*) + \mu^{*T} \nabla \boldsymbol{g}(\boldsymbol{x}^*) = 0^T \text{, and} \tag{77}$$

$$\mu^{*T} \boldsymbol{g}(\boldsymbol{x}^*) = 0 . \tag{78}$$

Figure 15. Definition of the Karush-Kuhn-Tucker theorem.

represents the derivative operator with respect to the independent variable x. λ and μ are the Lagrange multipliers associated with the constraints. Also note that multiple equality and inequality constraints (m and p in the above definition) can be present. This is accounted for by the vector notation for the Lagrange multipliers and the constraints.

Applying the KKT conditions to a simple optimization problem is helpful in understanding the optimization process. Suppose we wish to minimize the function,

$$f(\boldsymbol{x}) = (x_1 - 1)^2 + x_2 + 2 , \tag{79}$$

subject to the conditions,

$$h(\boldsymbol{x}) = x_2 - x_1 - 1 = 0 \text{, and} \tag{80}$$

$$g(\boldsymbol{x}) = x_2 + x_1 - 2 \leq 0 . \tag{81}$$

In other words, $h(x)$ defines the equality constraint and $g(x)$ defines the inequality constraint.

For all $\boldsymbol{x} \in \Re^2$, we have,

$$\nabla h(\boldsymbol{x}) = [-1, 1],\ \nabla g(\boldsymbol{x}) = [1, 1] \text{ and } \nabla f(\boldsymbol{x}) = [2x_1 - 2, 1]^{\cdot}.$$

Applying the KKT conditions from (76), (77) and (78), we get,

$$\mu \geq 0,$$

$$[2x_1 - 2 - \lambda + \mu,\ 1 + \lambda + \mu] = \mathbf{0}^T \text{ and}$$

$$\mu(x_2 + x_1 - 2) = 0.$$

If we let $\mu > 0$, then,

$$x_2 + x_1 - 2 = 0,$$

$$1 + \lambda + \mu = 0,$$

$$2x_1 - 2 - \lambda + \mu = 0, \text{ and}$$

$$x_2 - x_1 - 1 = 0.$$

Solving the above set of equations, we get,

$$x_1 = \frac{1}{2},\ \ x_2 = \frac{3}{2},\ \ \lambda = -1,\ \mu = 0.$$

This solution contradicts the assumption that μ is strictly positive. Therefore, the correct solution has to be obtained by letting $\mu = 0$. Then, the set of simultaneous equations

that need to be solved to obtain the optimal parameters are,

$$2x - 2 - \lambda = 0,$$

$$1 + \lambda = 0, \text{ and}$$

$$x_2 - x_1 - 1 = 0.$$

The solution to the above set of equations is,

$$x_1 = \frac{1}{2}, \quad x_2 = \frac{3}{2}, \quad \lambda = -1, \quad \mu = 0.$$

This solution satisfies the assumption that $\mu = 0$ as well as the constraints defined by g and h. Hence this is a valid solution to the problem.

In the SVM optimization process, using the third of the KKT conditions with (69) of the hyperplane optimization problem, we get

$$\alpha_i(y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1) = 0. \tag{82}$$

The above equation is useful in that it states that α_i is non-zero only for examples that satisfy,

$$y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) = 1. \tag{83}$$

As described earlier these examples are called the *support vectors*. This requirement also helps the optimization process in identifying examples that violate the KKT conditions. Identifying such vectors helps in speeding up optimization process as well as enables handling large datasets efficiently. Note that (71) and (82) are also the solutions for $\boldsymbol{w}$ and b respectively which completely define the optimal hyperplane.

Thus far we have seen the case where the training data is completely separable using a linear margin. However, we know that this is not the case with most real-world data. Classification problems typically involve non-separable data. Given such a training set, we still need to estimate the classifier that maximizes the margin and minimizes the errors on the training set. Optimization in such situations is typically accomplished by the use of *soft decision* classifiers where the classification of an example is tagged with a probability. However in the case of optimal margin classifiers, we use the concept of slack variables to find the optimal solution. Fig. 16 shows a simple hyperplane classifier with two training errors. The optimal-margin classifier can be extended to this non-separable case by using a set of slack variables that account for training errors. In this situation, the

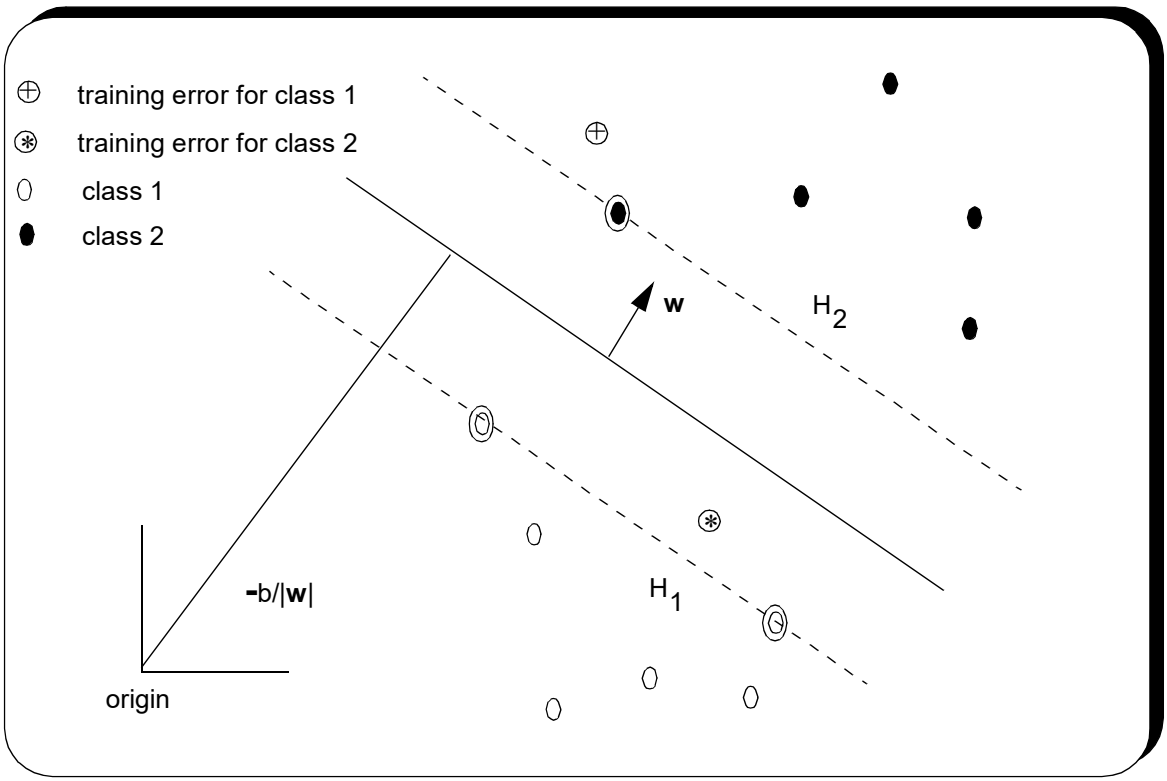

Figure 16. Examples of a soft-margin classifier which is robust to training errors.

inequality constraints to be satisfied by the hyperplane become,

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \geq +1 - \xi_i \qquad \text{for } y_i = +1, \tag{84}$$

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \leq -1 + \xi_i \qquad \text{for } y_i = -1, \text{ and} \tag{85}$$

$$\xi_i \geq 0 \qquad \forall i, \tag{86}$$

where ξ's are the slack variables. Now, the problem of finding the hyperplane which has a minimum number of training errors can be solved by minimizing the objective function defined as,

$$C\sum_i \theta(\xi_i) + \frac{1}{2}\|\boldsymbol{w}\|^2,$$

subject to the above described inequalities. In the above functional, θ is any function that measures the cost of a training error and C is the penalty for a training error. As a simple example, θ could be defined such that,

$$\theta(\xi) = 0 \text{ if } \xi = 0, \ \theta(\xi) = 1 \text{ if } \xi > 0. \tag{87}$$

Therefore, $\sum_i \theta(\xi_i)$ is an upper bound on the number of allowable training errors. The above problem is NP-complete [110]. Therefore, instead of using this formulation, we use an approximation and define the objective function as,

$$C\sum_i (\xi_i)^\sigma + \frac{1}{2}\|\boldsymbol{w}\|^2.$$

Typical values of $\sigma = 1$ or $\sigma = 2$ are used in practice. The solution of the new optimization now includes the constraint,

$$0 \leq \alpha_i \leq C. \tag{88}$$

This constraint is added to limit the effect of outliers on the definition of the classifier. In fact, there is a good chance that all examples with multipliers at the upper bound C are outliers. This property is used to speed up the optimization process and will be discussed in a later section. The higher the value of C, the harder the optimization process will try to minimize training errors. However this could mean increased time for convergence and in some cases, a larger support vector set.

4.3. Non-linear Hyperplane Classifiers

In the previous section we have seen the problem of estimating linear classifiers for cases where data is both separable or non-separable. This, however, does not help us solve many real-world situations where the data warrants the need for non-linear decision surfaces. This section deals with extending the SVM paradigm to handle such cases.

The power of SVMs lies in transforming data to a high dimensional space and constructing a linear binary classifier in this high dimensional space. Fig. 17 illustrates this idea using a simple example. This is a 2-class problem in a 2-dimensional input space. The two classes can be separated by a decision region in the form of circle which cannot be modeled by classifiers based on PCA or LDA [41]. The data in this 2-dimensional space is transformed to a 3-dimensional space via a simple transformation,

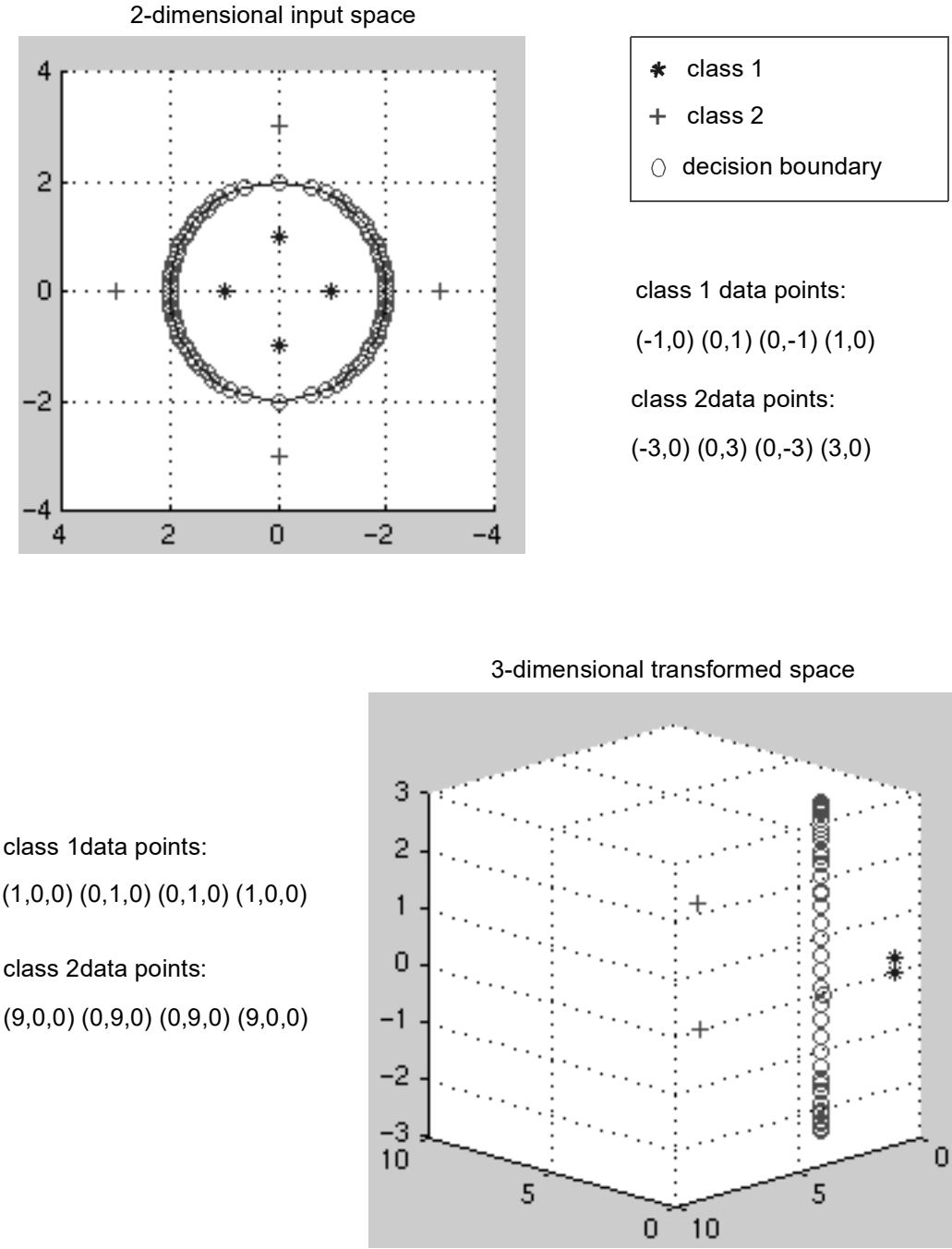

Figure 17. An illustration of the fact that the construction of a simple hyperplane in a higher dimensional space is equivalent to a non-linear decision surface in a lower dimensional space. In this example a decision surface in the form of a circle in a 2-dimensional space is modeled as a hyperplane in a 3-dimensional space.

$$(x, y) \Rightarrow (x^2, y^2, \sqrt{2}xy)\,. \tag{89}$$

From the figure it is clear that the two classes can be separated in the transformed space by defining a hyperplane passing through the points corresponding to the circular decision region. The following discussion formalizes this principle.

In all the formulations of the optimization in the previous sections notice that the only place the data points occur in the definition of the functional is the dot product. Suppose the data points are transformed to a higher dimension using

$$\Phi : \Re^n \rightarrow \Re^N, \tag{90}$$

where N is the dimensionality of the new feature space. In this new space we can still construct optimal margin classifiers with the only difference being that the simple dot product in (75) will now have to be replaced by $\Phi(\boldsymbol{x_i}) \cdot \Phi(\boldsymbol{x_j})$. It would be advantageous if we could define a "kernel" function K,

$$K(x_i, x_j) \;=\; \Phi(x_i) \cdot \Phi(x_j), \tag{91}$$

that could compute this dot product directly without knowing the explicit form of Φ. In this new formulation, the decision function will take the form,

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{x}, x_i) + b\,. \tag{92}$$

A function needs to satisfy certain conditions for it to be a valid kernel. The most important property the function must exhibit is that it needs to be a dot product in some

feature space. In order to affirm that a function does indeed represent a dot product in a higher dimensional space, Mercer's theorem can be used. This theorem ascertains that the pair $\{H, \Phi\}$ exists for a given kernel, where H defines the new feature space [28].

Mercer's Theorem

There exists a mapping Φ and an expansion

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum \Phi_i(\boldsymbol{x}) \cdot \Phi_i(\boldsymbol{y}) ,$$

if and only if, for any $g(x)$ such that

$\int g(\boldsymbol{x})^2 dx$ is finite,

then

$$\int K(\boldsymbol{x}, \boldsymbol{y}) g(\boldsymbol{x}) g(\boldsymbol{y}) dx dy > 0$$

If a kernel is a dot product in some feature space, Mercer's conditions must hold. Mercer's conditions do not however directly tell us how we define Φ or even the feature space H [27]. In reality, we do not have to explicitly have this information for constructing a classifier. However, knowing the transformation Φ can be useful in visualizing the new space.

Let us demonstrate this with a sample example. Suppose our data lies in a two dimensional space $\Re^2$ and we want to transform to a new space $\Re^3$. We can choose a Kernel, $K = (\boldsymbol{x} \cdot y)^2$. For this problem it is easy to find a mapping, Φ, from the input

space to the new transformed space such that,

$$(\boldsymbol{x} \cdot \boldsymbol{y})^2 = \Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{y}) . \tag{93}$$

If we define $\boldsymbol{x}$ as $\{x_1, x_2\}$ and $\boldsymbol{y}$ as $\{y_1, y_2\}$, then,

$$(\boldsymbol{x} \cdot \boldsymbol{y})^2 = x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 x_2 y_1 y_2 . \tag{94}$$

If we define the function Φ as,

$$\Phi(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix}, \tag{95}$$

we can see that equation is indeed true. This example demonstrates that it is possible to use kernels to transform data implicitly to a higher dimensional space.

Some of the other commonly used kernel functions are:

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y} + 1)^d \text{ — polynomial with degree } d \tag{96}$$

$$K(\boldsymbol{x}, \boldsymbol{y}) = Sigmoid[s(\boldsymbol{x} \cdot \boldsymbol{y}) + c] \text{ — sigmoid} \tag{97}$$

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp\{-\Upsilon |\boldsymbol{x} - \boldsymbol{y}|^2\} \text{ — radial basis function} \tag{98}$$

RBF kernels have been found to be the most powerful amongst the above mentioned kernels. However, convergence for RBF kernels takes longer than for the other kernels. RBF kernels can model closed decision surfaces unlike polynomial kernels. This property is extremely useful in classifying datasets where one class is completely encapsulated by data from another class.

4.4. SVM Training Process

Though a solution for the quadratic optimization of the functional in (74) is guaranteed based on the KKT conditions, the number of computations required can be very high depending on the separability of the data and the number of training data points. The reason that optimizing for all the data points simultaneously is nearly impossible computationally is the need to store all the possible dot products, $x_i \cdot x_j$, in order to perform the optimization. Instead of storing the matrix of these dot products we can compute them on an as-needed basis. However this can be computationally very expensive. For example, consider a 13-dimensional input space. Let the number of training examples be $20,000$. This means that in every iteration of the optimization process, $20000 \times 13 \times 20000$ multiplications need to be performed. Using the symmetry of the matrix, we can cut that number in half. However, this is still is a significant number of operations to be performed.

Several heuristics need to be considered to make SVM training feasible for large scale problems such as speech recognition. In this section, we discuss some significant modifications to the optimization problem formulation that make handling tasks with thousands of training points and support vectors possible. Most algorithms involving efficient and compact optimization are based on the premise that the global solution can be obtained (or approximated) by solving smaller problems at any given point in time. These algorithms mainly differ in the mechanisms used to define the smaller sub-problems. In the following section one such algorithm, *Chunking,* will be described.

Chunking

Chunking is based on the idea of dividing the optimization problem into sub-problems whose solution can be found efficiently. This method divides the training data into chunks and optimizes the functional for each chunk. Osuna proves that the chunking algorithm does in fact give the same solution as a global optimization process but takes much less operating memory and time [115].

The following discussion will use a vector notation of the optimization process for compactness of representation. Let us define a matrix Q, whose elements are,

$$Q_{ij} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{99}$$

Let the set of Lagrange multipliers be represented by the vector $\Lambda = [\alpha_0, \alpha_1, \alpha_2 \ldots]$. Let C be the penalty for misclassifying a training example and let $\boldsymbol{y}$ be the vector representing the class membership of the training examples. The functional to optimize, defined in (74), and the constraints on the Lagrange multipliers can then be written in a vector form as:

$$W(\Lambda) = -\Lambda \cdot \mathbf{1} + \frac{1}{2}\Lambda \cdot Q\Lambda, \text{ subject to,} \tag{100}$$

$$\Lambda^T \cdot \boldsymbol{y} = \mathbf{0} \tag{101}$$

$$\Lambda - C\mathbf{1} \leq \mathbf{0} \text{ and} \tag{102}$$

$$-\Lambda \leq \mathbf{0}. \tag{103}$$

This optimization needs to satisfy the KKT conditions to guarantee optimality. For

constraints represented by (101), (102) and (103), let the KKT multipliers be μ, Υ and Π. The multipliers for the two inequalities above have been separated for clarity though they were represented by the same variable in the definition of the KKT theorem. These multipliers are represented in a vector form for compactness. Each element of the vector is a multiplier corresponding to an inequality associated with an input sample. Note that the KKT conditions for this problem, based on (76), (77) and (78), are

$$\begin{aligned} \nabla W(\Lambda) + \Upsilon - \Pi + \mu \boldsymbol{y} &= \mathbf{0} \\ \Upsilon^{\boldsymbol{T}}(\Lambda - C\mathbf{1}) &= \mathbf{0} \\ \Pi^{\boldsymbol{T}}\Lambda &= \mathbf{0} \\ \Upsilon &\geq \mathbf{0} \\ \Pi &\geq \mathbf{0} \end{aligned} \tag{104}$$

We can simplify the above conditions to a simple form based on the range into which a Lagrange multiplier λ_i falls.

1. $0 < \lambda_i < C$: Since λ_i is non zero, π_i should be zero in order to satisfy the KKT conditions. Since λ_i is not equal to C, ν_i is equal to zero too. This results in the following equation.

$$(Q\Lambda)_i - 1 + \mu y_i = 0. \tag{105}$$

The product $(Q\Lambda)_i$ is the derivative of $\Lambda \cdot Q\Lambda$ with respect to λ_i and can be defined as,

$$(Q\Lambda)_i = \sum^{N} \lambda_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{106}$$

From earlier discussions, we know that the estimated label for x_i is given by,

$$g(\boldsymbol{x}_i) = \sum^{N} \alpha_j y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b = y_i \ . \tag{107}$$

Comparing (106) and (107), $(Q\Lambda)_i = y_i(y_i - b) = 1 - by_i$. Comparing with (105), we get $b = \mu$ under optimal conditions.

2. $\lambda_i = C$: From the third condition in (103), since λ_i is non-zero, π_i is zero, leading to

$$(Q\Lambda)_i - 1 + \nu_i + \mu y_i = 0 . \tag{108}$$

From (106), $(Q\Lambda)_i = y_i(g(\boldsymbol{x}_i) - b)$. Therefore,

$$y_i g(x_i) - y_i b - 1 + \nu_i + \mu y_i = 0$$

Since $b = \mu$ under optimal conditions,

$$y_i g(\boldsymbol{x}_i) = 1 - \nu_i .$$

However, since ν_i is required to be positive in order to satisfy the KKT conditions,

$$y_i g(\boldsymbol{x}_i) \leq 1 . \tag{109}$$

As defined earlier, the support vectors that have their multipliers at C are called *bounded support vectors* (BSVs) and are important in learning the inherent overlap in the data.

3. $\lambda_i = 0$: From the second equation in (104), this case implies that ν_i is zero and hence, $(Q\Lambda)_i - 1 + \pi_i + \mu y_i = 0$ Following a procedure similar to the previous

case it can be shown that for an example whose Lagrange multiplier equals zero,

$$y_i g(\boldsymbol{x}_i) \geq 1 . \tag{110}$$

The conditions defined in inequalities (109) and (110) are essential in deciding whether a multiplier is violating the KKT conditions. This is useful in choosing the best subset or "chunk" to optimize to speedup the overall optimization process. Suppose we define the working set as W and the non-working set (whose multipliers do not change while solving the sub-problems) as F. The Chunking algorithm can be specified in three simple steps [115]:

1. Choose $|W|$ training points from the data set at random.
2. Solve the optimization problem defined by the set W.
3. For some $j \in F$, such that:

   - $\lambda_j = 0$ and $y_j g(\boldsymbol{x}_j) < 1$
   - $\lambda_j = C$ and $y_j g(\boldsymbol{x}_j) > 1$
   - $0 < \lambda_j < C$ and $y_j g(\boldsymbol{x}_j) \neq 1$,

replace λ_i, $i \in W$, with λ_j and then solve the new minimization sub-problem given by

$$W(\Lambda_W) = -\Lambda_W^T \mathbf{1} + \frac{1}{2}\Lambda_W^T \cdot Q_{WW}\Lambda_W^T + \Lambda_W^T q_{WF}, \text{ subject to,} \tag{111}$$

$$\begin{aligned} (\Lambda_W^T \cdot \boldsymbol{y}_W + \Lambda_F^T \cdot \boldsymbol{y}_F = 0) \\ \Lambda_W - C\mathbf{1} \le 0 \\ -\Lambda_W \le 0 \end{aligned} \tag{112}$$

where,

$$(\boldsymbol{q}_{WF})_j = y_j \sum_{i \in F} \lambda_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{113}$$

The above algorithm is guaranteed to strictly improve the objective function [26]. The convex quadratic form of the objective function also guarantees that the algorithm will reach the global optimum solution within a finite number of iterations. The simplistic approach to implement this algorithm would be to add all examples from the non-working set that violate the KKT conditions to the working set. However, this can quickly become intractable especially when dealing with several thousand training examples. The challenge then is to find the best working set at each iteration and to devise heuristics to avoid redundant computations. The size of the working set can be kept under control by adding only examples that significantly violate the KKT conditions to the optimization process during each iteration. The work by Joachims addresses this issue of the SVM training algorithm [70,71]. The next few sections describe the algorithms implemented for the SVM toolkit developed by Joachims and used in this research.

The key to the success of the chunking algorithm is the choice of the working sets, B, that drive the optimization process towards convergence in fewer iterations. Though we can replace one vector in the working set during each iteration, when optimizing with a

large training set, choosing large chunks is preferable. *SVMLight* toolkit chooses the working set based on the algorithm proposed by Zoutendijk in his work on methods for feasible directions [116]. The goal of any method for feasible directions is to find the direction that provides the largest rate of increase of the objective function.

Let $\boldsymbol{d}$ be the vector representing the steepest feasible direction of descent. The idea is to choose $\boldsymbol{d}$ such that only a small fixed number of the elements of $\boldsymbol{d}$ are non-zero. The training examples corresponding to the non-zero elements form the working set. Zoutendijk's method helps identify the examples that contribute most to the change in the objective function. Mathematically the problem can be written as,

$$\begin{aligned} &\text{maximize} \quad \boldsymbol{f}(\boldsymbol{a}) \\ &\text{subject to} \quad \mathrm{A}\boldsymbol{a} \le \boldsymbol{b} \end{aligned} \quad . \tag{114}$$

The solution to the above problem is given by solving a set of linear equations subject to some constraints as,

$$\begin{aligned} &\text{maximize} \quad \nabla \boldsymbol{f}^{\boldsymbol{T}} \boldsymbol{d} \\ &\text{subject to} \quad \mathrm{A}\boldsymbol{d} \le \boldsymbol{0} \\ &\qquad\qquad \|\boldsymbol{d}\| \le 1 \end{aligned} \quad . \tag{115}$$

Several forms of normalization can be used for the direction vector. However the toolkit uses the normalization given by,

$$d_i \in \{-1, 0, 1\} \ , \tag{116}$$

which makes the search for the optimal feasible direction approximate but tractable.

For the particular problem of choosing the optimal working set, we need to maximize the quantity,

$$V(\boldsymbol{d}) = \nabla W(\Lambda^t) \cdot \boldsymbol{d}, \tag{117}$$

subject to the constraints,

$$\boldsymbol{y}^T \boldsymbol{d} = \mathbf{0}, \tag{118}$$

$$d_i \geq 0 \qquad for \text{ i: } \lambda_i = 0, \text{ and} \tag{119}$$

$$d_i \leq 0 \qquad for \text{ i: } \lambda_i = C. \tag{120}$$

Constraints 119 and 120 arise from the observation that for examples with the Lagrange multiplier equal to zero, the classification is correct and the direction of the solution should not be changed. However for examples with the multiplier at its upper bound, the classification is incorrect and the direction of optimization needs to be reversed. Solving the optimization in (117) at each iteration of the chunking algorithm can be very expensive. However, a simple modification to the problem definition can make the optimization simpler and less expensive.

To better understand the process of finding the examples that become part of the best working set, note that $\nabla W(\Lambda^t)$ is related to the estimated label for an example as:

$$\nabla W(\Lambda^t) = g - (1 + b), \tag{121}$$

where g is the estimated label as defined in (107). Therefore, the optimization to find the optimal working set can be defined in terms of the estimated labels instead of the

differential. In the absence of any constraints we could simply choose the required number of examples with highest values of the differential. Let the norm of the non-zero elements of the vector **d** be equal to unity. From (118) we know that the working set will then be composed of examples such that the number of examples with sign matches between d_i and y_i will be equal to the number of examples with sign mismatches. For all examples that violate the KKT conditions and $d_i = -y_i$, the maximum contribution to the objective function in (117) is provided by the examples with a minimum value for the quantity $y_i g_i$. To qualify the above statement, consider all positive examples where, $y_i = 1$ and $d_i = -1$. Since this example violates the KKT conditions, $g_i < 0$. So the maximal contribution to the objective function in (117) is determined by:

$$p_k = \max_{i:y_i = 1}(-g_i) = \min_{i:y_i = 1}(y_i g_i) \tag{122}$$

Similarly for all negative examples with sign mismatches, the maximal contribution can be determined by:

$$p_k = \max_{i:y_i = -1}(g_i) = \min_{i:y_i = -1}(y_i g_i) \tag{123}$$

Equations (122) and (123) suggest that the product yg can be used to sort examples based on their contribution to the objective function used to determine the steepest feasible direction. Therefore, when the product yg is sorted in ascending order, all examples with sign mismatches between d_i and y_i will be in the top half of the list.

Using a similar argument, for all examples that violate the KKT conditions and $d_i = y_i$, the maximal contribution to the objective function is provided by the example identified as:

$$p_k = \max_i (y_i g_i) \tag{124}$$

This example will therefore fall at the bottom of the aforementioned sorted list. In summary, the product yg can be used to choose a good working set. The procedure would involve that the training examples be sorted according to $w_i = y_i g_i$ and the new working set be chosen such that:

- half the elements from the top of the sorted list where $0 < \lambda_i < C$ or $d_i = -y_i$ and (119) and (120) are satisfied
- the other half from the bottom of the sorted list where $0 < \lambda_i < C$ or $d_i = y_i$ and (119) and (120) are satisfied.

This process is very inexpensive as compared to the main optimization process. Several other heuristics including caching of kernel evaluations and identification of bounded support vectors make this particular implementation of the SVMs very efficient. There are other implementations of SVMs that are efficient. These implementations allow training classifiers using a large data sample. One such algorithm is the Sequential Minimal Optimization (SMO) developed by Platt [117]. The idea here is to eliminate the complex quadratic optimizers from the implementation by breaking down the problem to solving a two-point optimization which can be done easily without the use of a quadratic optimization package.

4.5. Relationship to Other Machine Learning Techniques

The past few sections have described the theory and implementation of SVMs. SVMs offer significant advantages over some of the other widely used classification schemes. However the choice of the classifier is application dependent. The ease of classifier estimation and the numerical complexity of classifier have to be addressed before a choice for a classifier is made. This section describes two important alternate approaches to machine learning — decision trees and neural networks and attempts to compare and contrast these techniques with SVMs.

A decision tree is a simple inductive learning structure [118,119]. Given an instance of an object or situation, which is specified by a set of properties, the tree returns a "yes" or "no" decision about that instance. Each internal node in the tree represents a test on one of those properties, and the branches from the node are labeled with the possible outcomes of the test. Each leaf node is a Boolean classifier for the input instance. Decision trees can learn non-linear decision surfaces by approximating the non-linearities with a piece-wise linear solution. Decision trees can be optimized using principles of ERM.

Decision trees have several advantages over more complex machine learning techniques like SVMs. Simplicity of design is by far their biggest strength. With decision trees, incorporating knowledge about the datasets is easier. They can also handle non-numeric features effectively. However, decision trees typically suffer from overfitting [118]. Better generalization is achieved by computationally expensive processes. Tree pruning using cross-validation is the most common scheme [120-122]. SVMs on the other hand offer better generalization in a more concrete framework.

Neural networks, like SVMs, can learn complex non-linear decision regions. The most common parameter estimation algorithms used to estimate the parameters of a neural network are based on ERM. This makes them susceptible to many training problems previously discussed including overtraining and convergence. The convergence of the gradient estimation techniques is typically slow (SVMs have a similar problem). The structure of a neural network (number of layers, number of nodes in each layer, connections between nodes) is generally decided before the training process begins. The quality of the network also depends on the initial estimates of the network weights. Approaches to initializing these networks tend to be heuristic [96] in nature. In comparison, with SVMs, apart from the kernel function itself, all other parameters are learned via the SRM principle. Recently, techniques have been developed to automate the choice of the kernel parameters in a data-driven fashion [123].

Neural network classifiers applied to speech recognition tend to be very compact in terms of parameter usage when compared to HMM classifiers or SVMs. Since neural networks are comprised of interconnected nodes performing similar tasks, they lend themselves well for concurrent implementations. This is not the case with SVMs. Neural networks can also be trained easily as 1-vs-N classifiers which is not the case with SVMs. SVM estimation as 1-vs-1 classifiers is significantly simpler than 1-vs-N classifiers. This results is the need to estimate several classifiers to solve an N-class classification problem. In computationally restricted environments this can be a serious impediment.

In conclusion, though SVMs are well-founded mathematically to achieve good generalization while maintaining a high classification accuracy, we need to consider issues

such as computation complexity and ease of implementation in order to choose the best classifier for a given application. In situations where accuracy and generalization are the only criterion for selection SVMs should be explored. However for computation and memory constrained problems, other techniques like decision trees and neural networks also need to be explored.

4.6. Summary

In this chapter we have reviewed the theory of support vector machines. The principle of structural risk minimization and its relationship to empirical risk minimization has been discussed. The control over the generalization offered by SRM is what makes an SVM a very powerful machine learning technique. The design and construction of maximum margin hyperplanes which form the core of SVM estimation was discussed. Several issues related to making SVM estimation practical were discussed, specifically those addressed by the *SVMLight* toolkit that has been used for all SVM experiments reported in this dissertation. The next chapter deals with an alternative framework under which HMMs and SVMs are integrated in a hybrid speech recognition system.

CHAPTER 5

HYBRID RECOGNITION SYSTEM ARCHITECTURE

The last three chapters provided a mathematical framework for HMMs and SVMs. We have seen some examples of the power of SVMs as a classification tool and elegant iterative training algorithms for the estimation of HMMs. HMMs provide a way to handle the dynamic nature of speech due to their state machine formulation. SVMs suffer from the fact that they are inherently static classifiers. The main contribution of this dissertation involves the development of a hybrid system that effectively integrates these two powerful technologies in a framework capable of handling extremely large recognition tasks. This chapter describes the hybrid speech recognition framework developed as part of this dissertation.

5.1. Hybrid Connectionist Systems

Hybrid connectionist speech recognition systems have been used for continuous speech recognition for over a decade [34,48,51,52,53]. The motivation for looking at alternate techniques to the more prevalent HMM-based systems has been described in detail previously. The standard ML estimation criterion for HMM parameter estimation does not guarantee better classification and involves an assumption about the form of the underlying probability distribution. The assumption about the independence of features

across frames leads to a poor model for continuous speech. The first-order Markov assumption makes it hard to model duration. In addition to these problems, the power of neural networks to model complex decision regions discriminatively has been the motivation to explore connectionist speech recognition systems such as neural networks. Though neural network technology has progressed significantly, their application to dynamic patterns like speech has only been marginally successful [48,50] (though the gap has narrowed in recent times [52,53]). Most connectionist hybrid systems rely on an underlying HMM architecture to model the temporal evolution of speech.

One of the first successful implementations of a connectionist speech recognition system was the Connectionist Viterbi Training (CVT) system [124]. In this system, an initial frame-to-phoneme alignment was produced using an HMM-based system. The CVT procedure was then used to reestimate the output distributions associated with the HMMs. This is significantly different from other approaches where the neural networks were used as classifiers and the network outputs modeled the posteriors. The outputs of the trained networks in the CVT paradigm were used directly as HMM output probabilities.

After the first pass of network estimation, the process is repeated except that the alignments were now obtained using HMMs whose probability densities were modeled with neural networks. A cross-validation procedure was used to test for convergence. The neural network architecture used in this system was a recurrent network where history of the internal hidden states of the network was fed as input to the networks along with the current frame of speech data (similar to the architecture in Fig. 11). This system used a

fixed length time window as input to the networks. This is counter-intuitive based on our knowledge of sub-word units which have variable durations based on a number of linguistic effects. Another problem associated with this approach is the need for segment-level labeled data in order to train the networks. Each fixed length segment in the training data needs to be labeled by a baseline HMM system prior to training the networks. This is a very expensive process and needs a good baseline HMM system for accurate labeling.

Improvements to the above system were made by integrating time-alignment into the neural network estimation procedure. For this purpose, multi-state TDNNs were used as the core computational element in the network instead of a recurrent structure [125]. A simple TDNN is shown in Fig. 10. A multi-state TDNN differs from a conventional TDNN in that an additional layer that implements a simple DTW procedure is incorporated. Because of the DTW layer, precise segmentations are not required and scores can be accumulated every frame instead of every segment. The DTW layer has the effect of reducing the sensitivity of the system to frame-level classification errors. The networks themselves are trained using individually-labeled word segments. Recognition, however, does not require operation on segments of data.

The basic TDNN system has to be enhanced with several features including specialized networks that focus on word boundary detection and duration constraints to achieve good performance. Due to this complexity, most contemporary connectionist systems have reverted to using neural networks to estimate posterior probabilities and use the HMM structure to model temporal evolution [49,52,53]. The hybrid system

architecture developed in this dissertation is motivated by the success of such hybrid systems.

5.2. Posterior Estimation

Equation (125) defines the essential operation that the decoder portion of a conventional statistically-based speech recognition system performs. Our goal is to find the most likely word sequence. Depending on the acoustic models that constitute the word, this can be interpreted as finding the most likely model sequence,

$$\hat{M} = \underset{M}{\text{argmax}}\ p(A/M)p(M)\ , \tag{125}$$

where M is the acoustic model and A is the acoustic data. With HMMs, the class conditional probability $p(A/M)$ is obtained by evaluating Gaussian statistical models trained on groups of similarly labeled feature vectors. In connectionist systems, a neural network estimates the posterior, $p(M/A)$, and these posteriors are converted to likelihoods using Bayes' rule,

$$P(A/M) = \frac{P(M/A)P(A)}{P(M)}\ . \tag{126}$$

In the above equation, since the classifier estimates the posteriors directly, if we assume the effect of $P(A)$ to be insignificant for recognition, simply dividing the posterior with the *a priori* probability of the model M gives the required conditional probability (scaled likelihood) $P(A/M)$ that can be used for recognition [53]. A simple estimate of the prior probability is the relative frequency of the class as determined by the

class labels generated during Viterbi alignment of the training data. The assumption that the relative class frequencies in the training set are a good approximation for the statistics of the test set is a significant issue.

The priors can also be estimated based only on the phone frequencies in the lexicon defining the vocabulary of the dataset. These priors will encode some information regarding the pronunciation of the words in the vocabulary. In general, there is a mismatch between priors estimated as suggested above and the priors based on simple relative frequency of occurrence. This problem is however ignored in this work and equiprobable class priors are used throughout.

SVMs, by definition, provide a distance or discriminant which can be used to compare classifier outputs and arrive at a classification as described by (75). This is unlike neural networks whose output activations are in fact estimates of the posterior class probabilities [46]. This makes classification a process of choosing the class with the largest posterior probability. Posteriors are more directly related to classification than the class conditional probabilities (which are motivated by representational approaches). Also, the posterior can be converted to the class conditional probability as discussed above and can be used directly as part of the regular HMM framework.

On the other hand, one of the main concerns in using SVMs for speech recognition is that there is no clear relationship between the distance of the test pattern from the margin and the posterior class probability. If we can develop such a relationship, applying SVMs to speech recognition, within the HMM framework, is possible.

A crude estimate of the posterior probability can be obtained by modeling the posterior class probability distribution with a Gaussian. Another possibility is to use a simple histogram approach. The above methods however are not Bayesian in nature in that they do not account for the variability in the estimates of the SVM parameters. Ignoring this variability in the estimates often results in overly confident predictions by the classifiers on the test set [126]. Recent work on using moderated SVM outputs as estimates of the posterior probability has had success at the expense of increased computations [127]. We briefly discuss some of the methods that can be used to convert SVM distances to posterior probabilities.

The first option at hand is to use the SVM output directly:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b. \tag{127}$$

When $f > 0$, the test sample is classified as being in-class and when $f < 0$ the sample is classified as being out-of-class. In general f does not give any meaningful information and using f to break ties in a voting scheme is often counter-productive.

Another ad-hoc method is to use a clipped SVM output (e.g., clip scores greater than ± 1). This is a better approximation of the posterior. However, this output will show high confidence even in regions with low data density or regions that are far from the decision boundary.

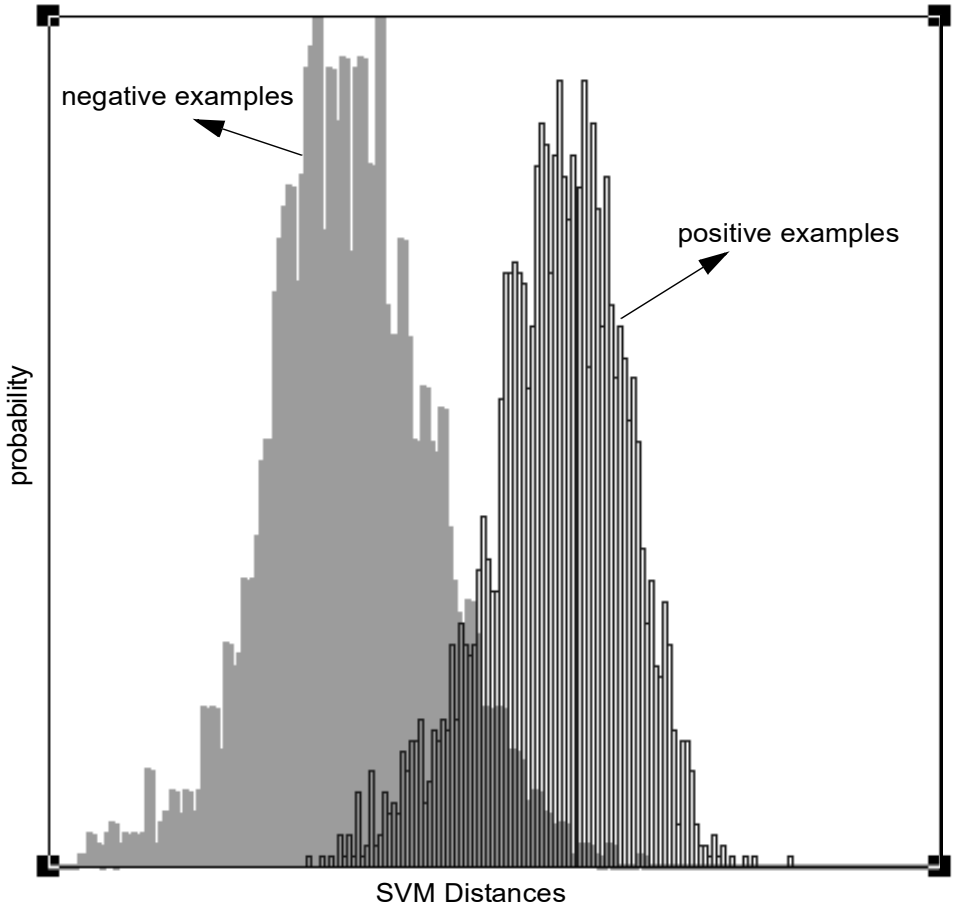

Figure 18. Histogram of SVM distances for positive and negative examples from the crossvalidation set.

A more interesting method is known as an unmoderated probability estimate based on maximum likelihood fitting. In this method, we estimate a sigmoid defined as,

$$p(y = 1/f) = \frac{1}{1 + \exp(Af + B)} . \quad (128)$$

Kwok's definitive work on moderated SVM outputs addresses the above issues in greater detail [127]. In the work reported in this dissertation, we have used the above formulation throughout.

Fig. 19 shows the distribution of the distances for positive and negative examples using a typical classifier. One possibility is to model these distance distributions using

Gaussians and then to compute the probability of the class given the SVM distance. Mathematically, this can be expressed as,

$$P(y = 1/f) = \frac{P(f/y = 1)P_1}{P(f/y = 1)P_1 + P(f/y = -1)P_{-1}} \quad , \tag{129}$$

where f is the SVM distance and y is the class label which takes the value ± 1. P_1 and P_{-1} are the in-class and out-of-class prior probabilities derived from the training data. Each of the class conditionals, $P(f|y)$ can be modeled as a Gaussian. Some simplifying assumptions can be made at this point without loss of generality.

Suppose we model each of the class-conditional probabilities with a Gaussian. Then,

$$P(f/y = 1) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp \frac{-(f-\mu_1)^2}{2\sigma_1^2} \quad , \tag{130}$$

and, (131)

$$P(f/y = -1) = \frac{1}{\sqrt{2\pi\sigma_{-1}^2}} \exp \frac{-(f-\mu_{-1})^2}{2\sigma_{-1}^2} \quad . \tag{132}$$

Without loss of generality, if we assume the Gaussians have equal variance we can write the posterior probability in (129) as,

$$p(y = 1/f) = \frac{P_1 P(f/y = 1)}{P_1 P(f/y = 1) + P_{-1} P(f/y = -1)} \tag{133}$$

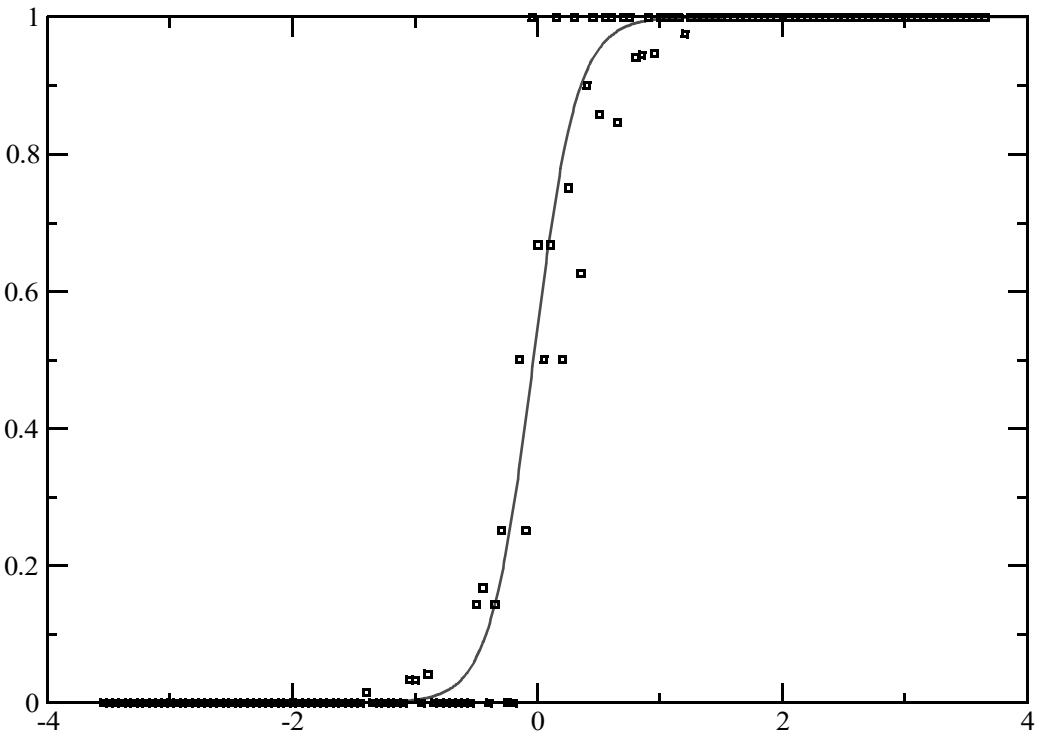

Figure 19. A sigmoid fit to the SVM distance-based posterior probability estimate.

$$= \frac{1}{1 + \frac{P_{-1}}{P_1}\exp\left(\frac{1}{2\sigma^2}((f-\mu_1)^2 - (f-\mu_{-1})^2)\right)} \tag{134}$$

$$= \frac{1}{1 + K\exp\left(\frac{1}{2\sigma^2}((\mu_1{}^2 - \mu_{-1}{}^2) + 2(\mu_{-1} - \mu_1)f)\right)} \tag{135}$$

$$\equiv \frac{1}{1 + \exp(Af + B)}, \tag{136}$$

where A and B are parameters that need to be estimated (using any suitable non-linear function estimator). An issue that arises from this formulation of estimating posteriors is that the distance estimates are heavily biased towards the training data. In order to avoid

biased estimates, a cross-validation set is used to estimate the parameters of the sigmoid [117]. The size of this data set can be determined based on the amount of training data that is available for the classifier. Fig. 19 shows the posteriors and the estimated sigmoid for a typical classifier.

5.3. Segmental Modeling

A logical step in building a hybrid system would be to replace a Bayesian classifier in a traditional HMM system with an SVM classifier that operates on acoustic feature vectors. However, the amount of training data and the confusion inherent in frame-level acoustic feature vectors is an issue worth addressing. Consider training a classifier to discriminate the phone ‘s’ from all other phones in a training set consisting of 40 hours of speech. At a frame rate of 100 frames per second, $14.4x10^6$ frames of data are available as training data for each classifier. Though efficient optimizers are available for SVM training, such a large amount of data could easily make the training process consume an inordinate amount of computational resources (on the order of months even on extremely fast processors). Another aspect of using frame-level data to train SVMs is the implicit assumption that the frame-level alignments that the HMM system generates are reliable. Experiments clearly indicate this to be a flawed assumption for conversational speech corpora such as SWITCHBOARD [15]. An iterative training procedure where the alignments are gradually improved is an option but is not addressed in this work [124,125].

In addition to the above implementation issues, there is clear evidence that it is extremely difficult to model human speech at the frame level where suprasegmental evidence such as duration cannot be used [54,55,56]. These issues serve as motivation to examine data at a coarser level. Segment-based approaches to modeling speech have been pursued in the past [57,58,59]. These approaches differ from traditional approaches in that they process one segment at a time instead of one frame at a time. The segments typically span several frames and can represent sub-word units motivated by phonology or abstract units derived by using data-driven techniques [128]. The motivation for most segment-based approaches is that the acoustic model needs to capture both temporal and spectral structure of speech which is clearly missing in frame-level classification schemes. Segmental approaches also overcome the assumption of conditional independence between frames of data in traditional HMM systems. Segmental data takes better advantage of the correlation in adjacent frames of data that is inherent in speech.

Despite their potential advantages, segment-based approaches have met with limited success. The inability to automatically generate reliable segmentations is a primary problem with this approach. This is often circumvented through the use of a hybrid architecture for acoustic modeling. The HMM paradigm provides an elegant framework to generate the most likely segmentations via a dynamic programming approach. The new classifier can then post process these segmentations to hypothesize the best word sequence.

Once the segmentation problem has been overcome, the next problem we face is the variable length or duration problem. Since segment duration is an important

speech-related feature, our classifier cannot simply discard this information. Resampling the variable length unit to form a uniform length feature vector has been pursued [58,59]. Resampling can be done both in the time domain or the feature domain. The problems associated with this approach include the need to interpolate data and the need to empirically determine the number of sample points. For example in a 15-frame phonetic segment, we could use frames 1, 5, 10, and 15 to comprise the segment-level feature vector. Using the same rule, a 12-frame segment could be composed of the frames 1, 4, 8 and 12. Elaborate quasi-linear sampling rules can be defined. It has been shown that sampling at naturally occurring boundaries is more effective than interpolation [58]. However these methods lack a clear experimental or mathematical motivation.

A simple but effective approach motivated by the 3-state HMMs used in most state-of-the-art speech recognition systems is to assume that the segments (phones in most cases) are composed of a fixed number of sections [129-131]. The first and third sections model the transition into and out of the segment, while the second section models the stable portion of the segment. We use segments composed of three sections in all recognition experiments reported in this work. This three part assumption fits well with our knowledge of phone articulation where the contextual effects cause a clear onset and offset for most phones, leaving a stable mid-section. This assumption may not be valid for certain consonants (e.g., plosives), where the stable section is very short or is non-existent. This flaw in modeling can be handled by incorporating a durational feature into each segment. We chose not to pursue this approach because of the problems with unreliable state-level alignments provided by the baseline HMM system.

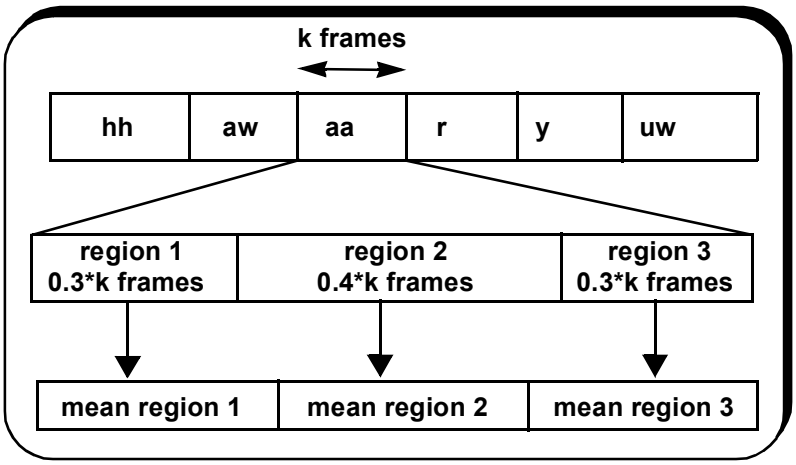

Figure 20. Composition of the segment level feature vector assuming a 3-4-3 proportion for the three sections.

Fig. 20 demonstrates the construction of a composite vector for a phone segment. SVM classifiers in our hybrid system operate on such composite vectors. The composite segment feature vectors are generated based on the alignments from a baseline 3-state Gaussian-mixture HMM system. The length of the composite vector is dependent on the number of sections in each segment and the dimensionality of the frame-level feature vectors. For example, with a 39-dimensional feature vector at the frame level and 3 sections per segment, the composite vector has a dimensionality of 117. SVM classifiers are trained on these composite vectors and recognition using the hybrid system is also performed using these segment-level composite vectors.

5.4. Modifications to an ASR System

The previous section described the mechanism used to generate likelihoods with an SVM classifier. The decoder in the ISIP ASR toolkit [78] was modified to replace Gaussian computations with SVM classifiers. This is the only significant change needed

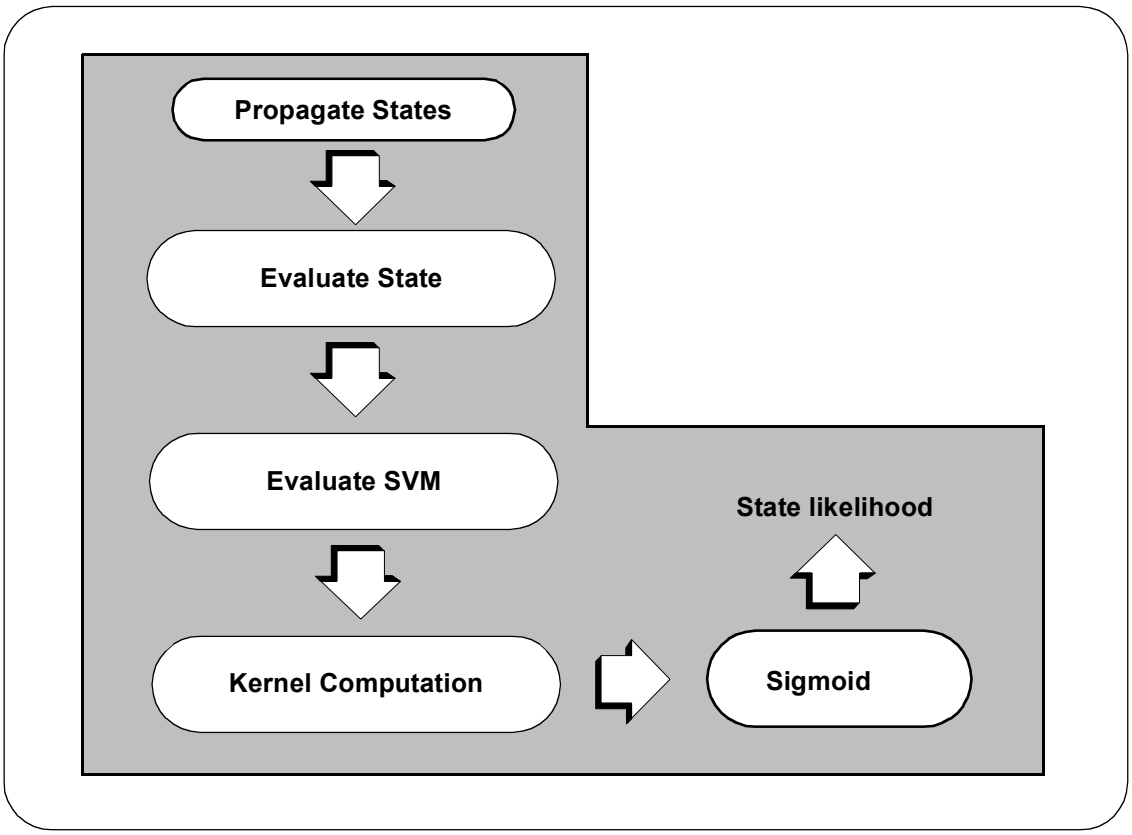

Figure 21. A schematic describing the process involved in computing likelihoods using SVMs in the hybrid system.

in the decoder to handle SVMs. Fig. 21 shows the process involved in generating state likelihoods in the hybrid decoder.

Several issues arise from using SVM-derived likelihoods in the decoding process. For an N-dimensional input feature vector, evaluating a Gaussian with a diagonal covariance matrix is equivalent to multiplying N Gaussians. The result is a product that has a very small value. However with SVM-derived likelihoods, the input feature vector is mapped to an SVM distance first and then the likelihood is estimated using the sigmoid. The range of the likelihood in this case are typically a couple of orders of magnitude larger

than those derived using HMMs. This requires changes to user-defined parameters such as the language model scaling factor and the word insertion penalty in the hybrid system.

5.5. N-best List Rescoring Paradigm

As a first step towards building a complex hybrid SVM/HMM system, we have explored a simple rescoring paradigm instead of an integrated approach often used in hybrid connectionist systems [132]. Assuming that we have already trained SVM classifiers for each phone in the model inventory, we generate N-best lists [97] using a conventional HMM system. These N-best lists can be processed in two ways. The first possibility is to use the segmentation from the best hypothesis generated by the HMM system and rescore all the other hypothesis using this segmentation. The lists can then be reordered and the new hypothesis can be chosen.

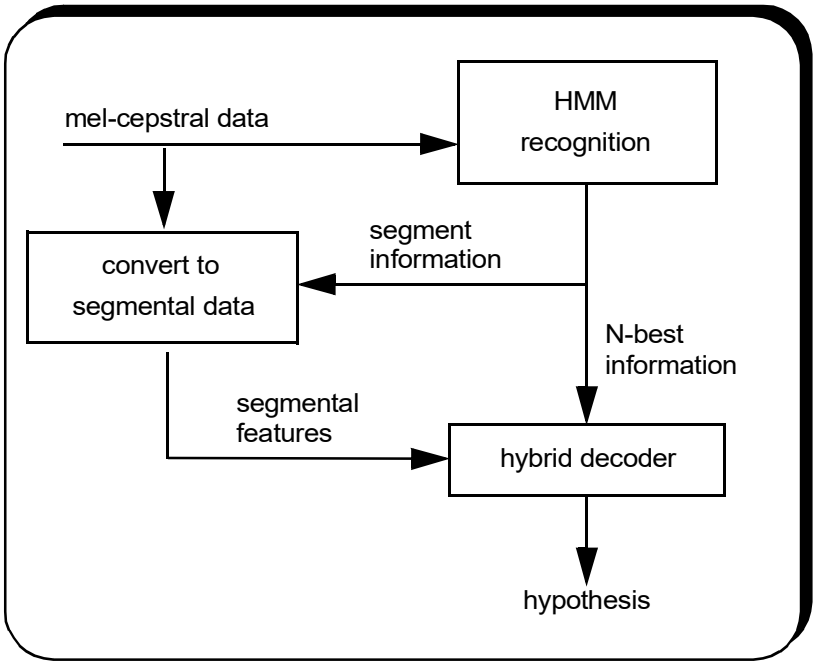

Figure 22. An N-best list rescoring paradigm which is the crux of the hybrid framework developed in this dissertation.

Another possibility is to generate a model-level alignment for each of the hypotheses in the N-best list using the HMM system. Based on these alignments, a segmentation for each hypothesis is generated. The likelihood of the corresponding hypothesis is computed by using SVMs to classify each segment. Posterior probabilities are computed using the sigmoid approximation discussed in the previous section. These probabilities are used to compute the utterance likelihood of each hypothesis in the N-best list. The N hypotheses can then be re-ranked using the new likelihoods and the best hypothesis can be chosen [76,77]. This scheme is shown in Fig. 22.

The two approaches described above differ significantly in several ways. Using a single segmentation to reorder the N-best list makes the hybrid recognition process simpler. A single pass of rescoring a word-graph comprised of the N-best hypothesis is sufficient to complete the rescoring process. However this approach does not conform to the methodology used for training the SVM classifiers where segmentation generated based on HMM alignments are used to train the classifiers. The second approach of using a separate segmentation to compute the likelihood of each hypothesis in the N-best list is well-matched to the training paradigm. This approach also fits well with approaches where segment graphs are used for decoding [129,133,134]. However, it is very cumbersome and computationally expensive. In this dissertation we analyze experimental results based on both approaches.

As a point of reference, we also produced results using a reference segmentation. These are a set of *oracle* experiments where the segmentations are produced by forced-alignments of the reference transcription. The results of these experiments provide

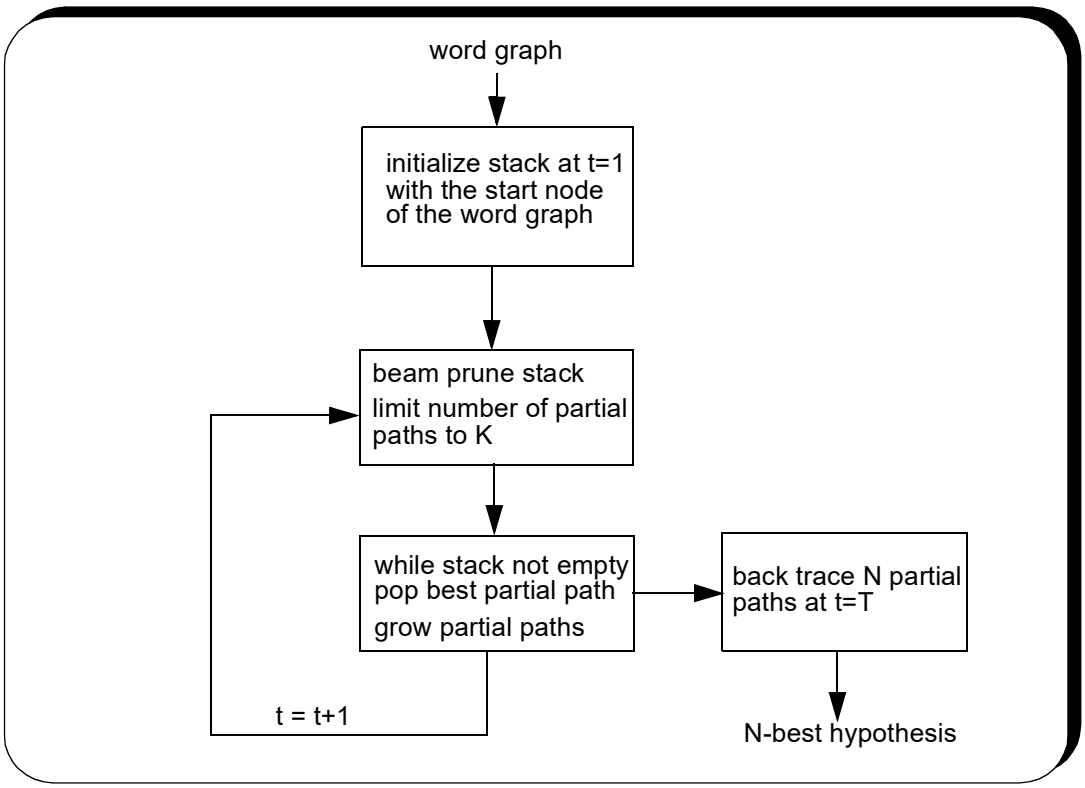

Figure 23. Flow-graph for the N-best rescoring paradigm.

a nice analysis tool as they give us a presumptive lower bound on the achievable error (the actual lower bound is the N-best list error rate, but it is a good assumption that we won't do better than a system with perfect knowledge of the reference segmentation). We hypothesize that this form of oracle experiment isolates the segmentation issue from the recognition process and hence calibrates the absolute improvements provided by the SVM classifiers.

5.6. N-best List Generation

N-best list generation is critical for the performance of the hybrid architecture explored in this research. Since the paradigm involves generating N-best lists using the

HMM system and then post processing these lists using the new classifiers, we assume that the N-best lists are rich enough to allow for any improvements over the baseline system. The ISIP ASR toolkit was enhanced to generate these N-best lists as part of the development of the hybrid system.

There are several ways in which N-best lists can be generated [135,136]. A* search is by far the most commonly used technique. However, since the ISIP ASR toolkit is capable of generating word graphs efficiently, we chose to implement a stack-based word graph to N-best list converter. Fig. 23 shows the flow graph for the word graph to N-best list conversion process. There are several issues that need to be addressed to make the N-best list generation efficient in terms of memory usage and execution time.

A single stack can be used for the complete process which contains partial paths possibly ending at various times. However care must be taken to normalize scores when partial paths are compared. In our implementation, a stack of partial paths per time frame is used. This makes the implementation straightforward with very little increase in bookkeeping overhead. This implementation also has an advantage in terms of applying pruning as the stack is built. A partial path is added to the stack only if it falls within the beam defined by the best scoring partial path at that time. This approach makes beam pruning very effective and speeds up the process by an order of magnitude. Our implementation also supports a fixed size for each of the stacks. The number of partial paths that are active at each point in time is limited by a user-specified parameter (denoted K).

Another issue that needs to be addressed is the fact that several paths in the word graph differ only in their time segmentations. For example, with $N = 10$, it could very well be that the list consists of only 2 or 3 unique word hypothesis. Since we propose to use the N-best lists in a rescoring paradigm, this redundancy is not useful. To account for this issue, the N-best list generation utility outputs only those hypotheses that have unique word sequences.

5.7. Summary

In this chapter we have described the hybrid SVM/HMM architecture developed as part of this dissertation. Several issues have been addressed that are specific to the use of SVMs in a hybrid system. Some of the key contributions include:

- Estimation of posterior probabilities based on SVM distances;
- Modifications to the ISIP ASR toolkit to accommodate SVM classifiers;
- Development of an N-best rescoring paradigm for the hybrid system;
- Implementation of an N-best list generation using a stack-based approach.

This hybrid architecture is an off-line processing mechanism and is boot-strapped using a baseline HMM system. Frame-level alignments are provided by a traditional HMM-based system.

CHAPTER 6

EXPERIMENTAL DATA AND BASELINE SYSTEMS

The primary goal of this dissertation is to evaluate the feasibility of using SVMs in conjunction with HMMs for continuous speech recognition. However, since systems that handle continuous speech are typically very complex and need to be tuned to the application, we begin by evaluating SVMs on simpler tasks. We gradually progressed to evaluation on continuous speech as we developed better insights on how to tune the classifiers. This chapter describes the datasets that have been used to evaluate SVM classifiers for this dissertation. Details such as vocabulary, pronunciations and dataset composition (both training and test data) are provided.

vowel	word	vowel	word
i:	heed	O	hod
I	hid	C:	hoard
E	head	U	hood
A	had	u:	who'd
a:	hard	3:	heard
Y	hud		

Table 1. The vowels forming the Deterding vowel dataset and the corresponding acoustic contexts in which they were collected. The vowel in context approach results in a more realistic articulation of each vowel.

6.1. Deterding Vowel Dataset

In our first pilot experiment, we applied SVMs to a publicly available vowel classification task: Deterding Vowels [137]. This was a good dataset to evaluate the basic SVM classifier since it has been used as a standard benchmark for several non-linear classifiers for several years. In this evaluation, speech data was collected at a 10 kHz sampling rate and low pass filtered at 4.7 kHz. The signal was then transformed to 10 log-area parameters, giving a 10 dimensional input space. A window duration of 50 msec. was used for generating the features. The training set consisted of 528 frames from eight speakers. The test set consisted of 462 frames from a different set of seven speakers. The speech data consisted of 11 vowels uttered by each speaker in a “h*d” context. Table 1 shows the vowel set and the corresponding words.

6.2. OGI Alphadigits

The performance of SVMs on the static classification of vowel data gave us good reason to expect the performance on continuous speech would be appreciably better than typical methods. Our initial tests of this hypothesis have been on a telephone alphadigit task [138]. Recent work on both alphabet and alphadigit systems has focused on resolving high error rates for easily confused words. In particular, the E-set (B, C, D, E, G, P, T, V, Z, THREE) and A-set (A, J, K, H, EIGHT) are most often used since these sounds have minimal acoustic differences between them. For instance, the letters B and D differ primarily in the first 10-20 ms during the consonant portion of the letter [139].

word	pronunciation	word	pronunciation
a	ey	s	eh s
b	b iy	t	t iy
c	s iy	u	y uw
d	d iy	v	v iy
e	iy	w	d ah b ax l y uw
f	eh f	x	eh k s
g	jh iy	y	w ay
h	eh ch	z	z iy
i	ay	one	w ah n
j	jh ey	two	t uw
k	k ey	three	th r iy
l	eh l	four	f ow r
m	eh m	five	f ay v
n	eh n	six	s ih k s
o, oh	ow	seven	s eh v ih n
p	p iy	eight	ey t
q	k y uw	nine	n ay n
r	aa r	zero	z iy r ow

Table 2. Lexicon used for recognition for the OGI Alphadigits dataset.

The OGI Alphadigit Corpus [140] is a telephone database collected from approximately 3000 subjects. Each subject was a volunteer responding to a posting on the Usenet. The subjects were given a list of either 19 or 29 alphanumeric strings to speak. The strings in the lists were each six words long, and each list was “set up to balance phonetic context between all letter and digit pairs” [140]. There were 1102 separate prompting strings which gave a balanced coverage of vocabulary items and contexts. The

training, cross-validation and test sets consisted of 51544, 13926 and 3329 utterances respectively, each balanced for gender. The data sets have been chosen to support speaker independent evaluations [141]. The alphadigits database has a vocabulary of 36 words — 26 alphabets and 11 digits including "oh". Table 2 shows the vocabulary and the pronunciations used to model this dataset.

As described in the previous chapters, we have developed a hybrid architecture for speech recognition using a combination of a traditional HMM system and SVM classifiers. To this end, we have developed a baseline HMM system which was used to generate the segmental data used to train and test SVM classifiers. The baseline system was also used to generate the N-best lists which were re-ordered using SVM classifiers to obtain the final hypothesis.

The HMM baseline used FFT-derived mel-cepstral features with cepstral mean normalization. All phone models are standard 3-state left-to-right models without skip states. These models are seeded with a single Gaussian observation distribution. A context-independent phone system is estimated using multiple passes of Baum-Welch estimation. A context-dependent phone system is then bootstrapped from the context-independent system. Four passes of Baum-Welch reestimation are used to generate single-component mixture distributions for the triphone models. At this stage of the estimation process, states are tied via a phonetic decision tree that is built using an ML formulation. The state-tied triphone models are then estimated using four more passes of Baum-Welch estimation. The number of Gaussians per state are then increased to 12 using

ph.	word	ph.	word	ph.	word	ph.	word	ph.	word
aa	bOdy	ch	CHurch	f	Four	m	Mine	sh	maSH
ae	Angry	d	Do	g	God	n	Novel	t	Two
ah	bUt	dh	THere	hh	Hello	ng	kiNG	th	THree
ao	fOssil	eh	gEt	ih	hIt	ow	hOme	uh	bOOk
aw	hOWl	el	cattLE	iy	hEAt	oy	bOY	uw	tOOl
ax	blossOms	en	buttON	jh	June	p	Pair	v	Very
ay	fIle	er	bURgER	k	Kill	r	Result	w	War
b	Be	ey	sAY	l	Low	s	Sing	y	Yellow
z	Zero	zh	uSual						

Table 3. Phone set used for the SWITCHBOARD recognition experiments. A total of 42 phones are used to model all pronunciations in the lexicon.

a standard divide-by-two clustering algorithm. This baseline system performs at 11.9% word error rate on the 3329 utterance evaluation test set for this corpus.

6.3. SWITCHBOARD

SWITCHBOARD is a large multispeaker corpus of telephone conversations collected in the early 90's with speakers representing a wide geographic distribution of the U.S.A. [15]. The database consists of 2,430 conversations and spans 500 speakers. A total of 240 hours of speech was collected. SWITCHBOARD is the most commonly used corpus to evaluate conversational speech recognition systems. The conversations pose several problems for recognition systems including the immense variability in speaking styles and the casual (poorly articulated) nature of the conversations. After nearly a decade of research the best systems achieve about 20% word error rate on this corpus [32,33].

We have used a subset of this corpus for training and testing systems for this dissertation. The training set consists of 60 hours of speech where the conversations were linguistically segmented after generating initial energy-based segments. The training set consists of 114,441 utterances while the development test set consists of 2427 utterances [93]. These utterances have an average length of six words and an average duration of two seconds. As with the other datasets used in this dissertation, this is a speaker independent evaluation. The test set vocabulary is approximately 22,000 words while the training set vocabulary has over 80,000 words. For all the phone-based systems reported in this dissertation, a 42-phone set has been used. The phones and their definitions are shown in Table 3.

The baseline HMM system was trained on 60 hours of SWITCHBOARD data from 2,998 conversation sides. The input features were mel-cepstral coefficients which had been normalized to have a zero-mean (side-based cepstral mean normalization was used [142]) and unit variance [143]. Using this data, context-independent phone models were trained iteratively starting from one mixture component to a final configuration of 32 mixture components per HMM state. These models were then used to generate phone-level alignments. The phone alignments were used throughout the remainder of our training process. Cross-word context-dependent phone models were seeded with single-mixture monophones, reestimated using a four pass procedure, and then clustered using phonetic decision trees [91]. Mixture splitting (terminating at 12 mixtures per state) was performed using an iterative splitting and training scheme [144]. This baseline system has a word error rate of 41.6% on the development test set.

6.4. Summary

In this chapter we have described the datasets used to evaluate the SVM classifiers developed as part of this research. These datasets include a standard non-linear classification task — Deterding Vowel Data and two continuous speech recognition tasks. The OGI Alphadigits Corpus is a small vocabulary task which is good dataset to assess the improvements SVMs provide on acoustic modeling, since it contains many sounds that are acoustically very similar (minimal pairs). This dataset is used to analyses several aspects of the hybrid system including the effect of parameter tuning. The baseline for this task using a traditional cross-word context-dependent HMM system is 11.9% on the evaluation set. The SWITCHBOARD database is used to evaluate the hybrid system on a challenging large vocabulary task. The baseline HMM system performs at 41.6% word error rate on this task. The baseline systems for both the continuous speech recognition tasks have been briefly described.

CHAPTER 7

EXPERIMENTS

The previous chapters described the theory and implementation of a hybrid SVM/ HMM system. In this chapter, we analyze results obtained on the databases described in chapter (6). We also present a detailed discussion of the classifier design in terms of data distribution and parameter selection. The performance of the SVM classifiers and the hybrid system are compared to the baselines used in this dissertation.

7.1. Deterding Vowel Data

The Deterding vowel dataset is one of the most widely used for benchmarking non-linear classifiers. The small training set and significant confusion in the vowel data make it a very challenging task. The best neural network classifiers (Gaussian Node Network) produce a misclassification rate of 44% [50]. In all experiments, for each class, a one-vs-all classifier was estimated and the final hypothesis was generated using a simple voting scheme where we choose the class which has the largest raw SVM output.

7.1.1. Effect of RBF Kernel Parameters on Classification

A traditional method for estimating an RBF classifier involves finding the RBF centers for each class separately using a clustering mechanism such as K-MEANS [41].

gamma (C=10)	classification error %	C (gamma=0.5)	classification error %
0.2	45	1	58
0.3	40	2	43
0.4	35	3	43
0.5	36	4	43
0.6	35	5	39
0.7	35	8	37
0.8	36	10	37
0.9	36	20	36
1.0	37	50	36
		100	36

Table 4. Effect of the kernel parameters on the classification performance of RBF kernel-based SVMs.

Next, we estimate weights corresponding to each cluster center to complete the definition of the classifier. However, the goal of the optimization process used to compute the weights is typically not improved discrimination, but better representation. The training process also requires using heuristics to determine the number of cluster centers. On the other hand, the SVM approach for estimating RBF classifiers is more elegant where the number of centers (support vectors in this case) and their weights are learned automatically in a discriminative framework.

The parameters of interest in tuning an RBF kernel are Υ, the variance of the kernel as defined in (98) and C, the parameter used to penalize training errors during SVM estimation. Table 4 shows the performance of SVMs using RBF kernels for a wide range of parameter values. The results clearly indicate that the performance of the

classifiers is very closely tied to the parameter setting, though there exists a pretty wide "sweet-spot." Another interesting observation is the effect of C on the performance. Note that for values of C greater than 20 the performance does not change. This suggests that a penalty of 20 has already accounted for all overlapped data and a larger value of C will have no additional benefit.

7.1.2. Effect of Polynomial Parameters on Classification

The polynomial kernel has only one free control parameter, the order of the polynomial. Typically, with increasing polynomial order, more complex decision regions can be modelled, though this requires more support vectors. Table 5 demonstrates the variation of performance as a function of the polynomial order. It is interesting to see how the number of support vectors increases as the classifiers increase in complexity. Though we would expect that the performance of the system improves with polynomial order, we conjecture that a small amount of training data is insufficient to estimate the higher order polynomial kernel parameters effectively.

order	classification error %	average number of SVs per classifier
2	49	35
3	52	45
4	52	52
5	56	55

Table 5. Performance of a polynomial kernel as a function of the polynomial order.

Performance using the RBF and polynomial kernels (35% and 49% respectively) is better than most nonlinear classification schemes [46]. Other SVM implementations have reported results similar to the numbers reported in this dissertation [75]. The best performance reported on this data set is, however, 29% error using a speaker adaptation scheme called Separable Mixture Models [145]. The performance obtained on this task demonstrates the power of SVMs in challenging classification tasks, provided that sufficient training data exists. The above experiments also a give us a sense of the importance of parameter tuning.

7.2. OGI Alphadigits

Building classifiers for a continuous speech recognition task is very different and a far more complex process than building classifiers for static classification tasks such as the Deterding vowel dataset. The number of tokens available for each classifier is typically in the tens of thousands. The number of classifiers is not uniquely determined by the task, but has to be decided based on the system complexity and data availability. The following sections provide the background for the design decisions made in developing our hybrid system.

7.2.1. Classifier Design

Thus far we have not addressed a fundamental issue in classifier design — should the classifiers be one-vs-one or one-vs-all? As the name suggests, one-vs-one classifiers learn to discriminate one class from an another class and one-vs-all classifiers learn to discriminate one class from all other classes. One-vs-one classifiers are typically smaller

and can be estimated using fewer resources than one-vs-all classifiers. When the number of classes is N we need to estimate $N(N-1)/2$ one-vs-one classifiers as compared to N one-vs-many classifiers. On several standard classification tasks it has been proven that one-vs-one classifiers are marginally more accurate than one-vs-many classifiers [146]. In the case of phone classifiers for the tasks at hand the number of one-vs-one classifiers that need to be estimated is significantly greater that one-vs-many classifiers — 406 for the OGI alphadigits database and 820 for SWITCHBOARD. Estimating these classifiers can be very time consuming. Therefore, we chose to use one-vs-many classifiers in all experiments reported here. The next issue we need to address is amount of data required to train each one-vs-many classifier.

7.2.2. Classifier Estimation

The baseline HMM system described in section 6.2 was used to generate segmented training data by Viterbi-aligning the training reference transcription to the acoustic data. The time marks derived from this Viterbi alignment were used to extract the segments. Before extraction, each feature dimension was normalized to the range [-1,1] to

Segmentation Proportions	WER (%) RBF kernel	WER (%) polynomial kernel
2-4-2	11.0	11.3
3-4-3	11.0	11.5
4-4-4	11.1	11.4

Table 6. Comparison of performance as a function of the segment proportions. 1-best hypothesis segmentations are used to generate the SVM segmentations and 10-best lists are rescored.

improve the convergence properties of the quadratic optimizers used as part of the SVM estimation utilities [70]. For each phone in the training transcription, a segment was extracted. This segment was divided into three parts as detailed previously in Fig. 20. An additional parameter describing the log of the segment duration was added to yield a composite vector of size 3 * 39 features + 1 log duration = 118 features. Once the training sets were generated for all the classifiers, the SVMLight [70] utilities were used to train each of the 29 phone SVM models.

In Table 6, performance as a function of three different segment ratios is presented. These experiments suggest that SVM classifiers are insensitive to the specific proportion that is chosen. This trend is observed consistently on both the regular experiments where N-best lists are rescored using HMM hypothesis-based segments and the *oracle* experiments where N-best lists are rescored using reference transcription-based segments.

7.2.3. Data Distribution

For each phone shown in Table 2, an SVM model was trained to discriminate between this phone and all other phones (one-vs-all models), generating a total of 29 models. In order to limit the number of samples (especially the out-of-class data) that is required by each classifier, a heuristic data selection process was used. Some of the important heuristics used included the requirement that the training set consists of equal amounts of within-class and out-of-class data. All within-class data available for a phone is by default part of the training set. The out-of-class data was randomly chosen such that one half of the out-of-class data came from phones that were phonetically similar to the

phone of interest and one half came from all other phones. The phonetic similarity sets that were used in this experiment are shown in Table 7. Balancing the data by similarity allowed for more data to be used during training. This scheme was used to train all classifiers for the continuous speech recognition tasks reported in this work.

An alternate mechanism for choosing data is to use a phone confusion matrix in a iterative framework. At the end of each training iteration, a confusion matrix can be generated for every classifier. The next iteration would include more data from a confusable phone as compared to a less confusable phone. This approach is similar in flavor to the concept of *Boosting* used in several machine learning techniques [147]. Another approach would be to use an agglomerative clustering scheme where classifiers are organized in the form of a tree [65]. The classifiers closer to the root node perform coarse classification (vowels vs. nasals for example) while the classifiers at the leaves perform fine-grain classification (‘m’ vs. ‘n’ for example).

set	phonemes
vowels	aa, ah, ax, ay, eh, ey, ih, iy, ow, uw
fricatives	ch, f, s, th, v, z
nasals	m, n
approximants	w, r, l, y
stops	b, d, jh, k, p, t

Table 7. Phonetic similarity sets used to build SVM training sets for the OGI alphadigits task.

RBF gamma	WER (%) hypothesis Segmentation	WER (%) Reference Segmentation	polynomial order	WER (%) hypothesis Segmentation	WER (%) Reference Segmentation
0.1	13.2	9.2	3	11.6	7.7
0.4	11.1	7.2	**4**	**11.4**	**7.6**
0.5	11.1	7.1	5	11.5	7.5
0.6	11.1	7.0	6	11.5	7.5
0.7	**11.0**	**7.0**	7	11.9	7.8
1.0	11.0	7.0			
5.0	12.7	8.1			

Table 8. Comparison of word error rates as a function of the RBF kernel width (gamma) and the polynomial kernel order. Results are shown for a 3-4-3 segment proportion with the error penalty, C, set to 50. The WER for the baseline HMM system is 11.9%.

7.2.4. Effect of Kernel Parameters on Performance

Table 8 shows the performance of the hybrid SVM system as a function of the kernel parameters. These results were generated with 10-best lists whose total list error (the error inherent in the lists themselves) was 4.0%. The list error rate is the best performance any system can achieve by postprocessing these N-best lists. Though we would ideally like this number to as close to zero as possible, the constraints placed by the limitations of the knowledge sources used in the system (acoustic models, language models etc.) force this number to be non-zero. In addition, the size of the N-best list has to be kept small for most post-processing operations to make the systems computationally attractive.

As with the vowel classification data, the RBF kernel performance was superior to the polynomial kernel. In addition, as we observed in the vowel classification task, the generalization performance is fairly flat for a wide range of kernel parameters. The 1-best

hypothesis segmentation was used to produce an 11.0% WER using an RBF kernel. To provide an equivalent and fair comparison with the HMM system we have rescored the 10-best lists with the baseline HMM system. Using a single segmentation to rescore the 10-best lists does force a few hypothesis in the lists to become inactive because of the duration constraints. The effective average list size after eliminating hypothesis that do not satisfy the durational constraints imposed by the segmentation is 6.9. The result for the baseline HMM system using these new N-best lists remains the same indicating that the improvements provided by the hybrid-system are indeed because of better classifiers and not the smaller search space.

As a point of reference, we also produced results (Table 8) using the reference transcription to generate the segments. Using this *oracle* segmentation, the best performance we get on this task is 7.0% WER. This is a 36% relative improvement in performance over the best configuration of the hybrid system using hypothesis-based segmentations. The effective average list size after eliminating hypotheses that do not satisfy the durational constraints imposed by the *oracle* segmentation is 6.7. This experiment shows that SVMs efficiently lock on to good segmentations. However, when we try to let the SVMs choose the best segmentation and hypothesis combination by using the N-best segmentations the performance gets worse (11.8% WER as shown in Table 9). This apparent anomaly suggests the need to incorporate variations in segmentation into the classifier estimation process. Relaxing this strong interdependence between the segmentation and the SVM performance is a point for further research.

Data Class	HMM (%WER)	SVM (%WER)	HMM+SVM (%WER)
a-set	13.5	11.5	11.1
e-set	23.1	22.4	20.6
digits	5.1	6.4	4.7
alphabets	15.1	14.3	13.3
nasals	12.1	12.9	12.0
plosives	22.6	21.0	18.9
Overall	**11.9**	**11.8**	**10.6**

Table 9. Comparison of performance of the HMM and SVM systems in isolation and in combination as a function of prominent word classes in the alphadigits vocabulary. Unlike the system used to generate Table 8, these numbers are generated by reordering N-best lists using N segmentations.

The goal in SRM is to build a classifier which balances generalization with discrimination on the training set. Table 8 shows how the RBF kernel parameter is used as a tuning parameter to achieve this balance. As gamma increases, the variance of the RBF kernel decreases. This in turn produces a narrower support region in a high-dimensional space. This support region requires a larger number of support vectors and leads to overfitting as shown when gamma is set to 5.0. As gamma decreases, the number of support vectors decreases, which leads to a smoother decision surface. Eventually, we reduce the number of support vectors to a point where the decision region is overly smooth (gamma = 0.1), and performance degrades. In agreement with the concept described in Fig. 12, these numbers show that the optimal operating point is a clear trade-off between the two extremes of overfitting (poor generalization) and oversmoothing (poor performance).

7.2.5. Likelihood Combination-Based Recognition

Error analysis often helps understand systems better. For the alphadigits task, we compared and contrasted the error modalities of the hybrid system and the HMM baseline system. The typical confusions at the word-level have been used to define the error modes. We use the same word class groups used in [139] which have been identified as the primary error modalities for this task.

Table 9 shows the performance of the hybrid system and the baseline HMM system on the error modalities being analyzed. The two systems seem to have complementary strengths to a large extent. For example, the SVM hybrid system is better at handling the "a-set" word-pairs better than the HMM system. However, the HMM system does better on nasals. This analysis led us to explore the benefits of combining the outputs of the two systems to produce the final hypothesis.

We explored a system combination scheme where the word-likelihood score from the SVM system was combined with the word-likelihood score from the HMM baseline according to

$$likelihood = \text{SVM score} + \frac{\text{HMM Score}}{\text{norm factor}}. \tag{137}$$

This method does require the estimation of another free parameter to normalize the respective scores. As the normalization factor increases, the likelihood is dominated by the SVM hypothesis. Likewise, as the normalization factor decreases, the HMM score dominates. Table 10 shows the results of this method for a sweep of normalization factors. A normalization factor of 200 delivers our best overall error rate of 10.6% WER. The last

Normalization Factor	HMM+SVM (%WER)
100000	11.8
10000	11.4
1000	10.9
500	10.8
200	**10.6**
100	10.7
50	10.8
0.0001	11.9

Table 10. Error rate as a function of the normalization factor. The optimal value is 200, and provides an error rate of 10.6%. When the normalization factor is 0.0001, in effect the system uses only the HMM-derived likelihoods.

column in Table 9 is generated using this normalization factor. Interestingly, this score combination scheme shows gains for every error modality explored in this recognition task. This fact is particularly encouraging and warrants further research.

7.3. SWITCHBOARD

The SWITCHBOARD (SWB) task is very different from the OGI alphadigits task in terms of acoustic confusibility and classifier complexity. The alphadigits task is simpler in the sense that most of the confusions occur as minimal pairs that can be categorized into a few error classes. However, in SWB, because of the conversational speaking style, recognition systems make more egregious errors. As will be discussed in a later section, this is an important factor that contributes to a wide variety in the segmentations for phones. This becomes even more important in the light of the findings reported in section 7.2.4 where the use of good segmentations seem to help the SVM classifiers

significantly improve performance. The baseline HMM system for this task performs at 41.6% WER.

7.3.1. Classifier Estimation and Data Distribution

We follow a similar classifier estimation process used for the OGI alphadigits task to train classifiers for the SWB task. We estimate 43 classifiers for this task. Since a standard cross-validation set does not exist for this task, we used 90,000 utterances from the 60 hours of training data to train the classifiers. The cross-validation set consisted of 24,000 utterances. In order to make the computational complexity feasible, we limited the number of positive examples for any classifier to 30,000. These examples were, however, chosen at random. Note that we take this approach with the sole purpose of reducing training time and not reduced variability of the inputs to the SVM classifiers. We do not explore other data selection schemes in this work, though using a clustering scheme could achieve better results than the random selection used here.

set	phonemes
vowels	eh ih ao ae ah aa uw uh er ay oy ey iy aw ow ax
fricatives	s sh z zh f v ch jh th dh
nasals	m n en ng
approximants	l el r w y hh
stops	b d k p t g

Table 11. Phonetic similarity sets used to build SVM training sets for the SWB task

The out-of-class data for each classifier was randomly chosen such that one half of the out-of-class data came from phones that were phonetically similar to the phone of interest and one half came from all other phones. The phonetic similarity sets used for this purpose are shown in Table 11. For all experiments conducted for this task, we used segments generated using a 3-4-3 proportion. This was based on the results for the alphadigits task that show that performance of the hybrid system is relatively insensitive to the proportion. Since the performance of RBF kernels has proven better than the polynomial kernels on all tasks reported in this work thus far, we chose to use only the RBF kernel for experiments on SWB.

7.3.2. Effect of Segmentation Source on Performance

We have seen the effect of using reference-based segmentations and hypothesis-based segmentations on the performance of the hybrid system for the alphadigits task. In this section we discuss the effect of segmentations on the performance of the hybrid system on the SWB task. We use 10-best lists with a list error rate of 29.5% for all experiments. Three distinct experiments were performed on this task in an effort to explore the effect of segmentation on system performance. This was motivated by the performance numbers we saw in Tables 8 and 9 which clearly show that segmentation is a major issue in the performance of the hybrid system.

In the first experiment we used a segmentation derived from the HMM's hypothesis to rescore the N-best list. This hybrid setup does improve performance over the baseline, albeit only marginally — 40.6% compared to a baseline of 41.6%. The second

experiment was performed by using N segmentations to rescore each of the utterances in the N-best lists. From the experimental results on the alphadigits task we expect the performance with this setup to be worse than 40.6%. The performance of the system did indeed get worse — 42.1% WER. When HMM models were used instead of SVMs under the same setup, the HMM system achieved a WER of 42.3% compared to the baseline of 41.6%. From this result we deduce that the lack of any language modeling information when we reorder the N-best lists is the reason for this degradation in performance.

The third experiment was the *oracle* experiment described in section 5.5. Using the reference segmentation to rescore the N-best list gave a WER of 36.1%. Once again, this improvement we see using the *oracle* segmentations shows that the segmentation issue needs further exploration.

7.4. Oracle Experiments

In this section we compare and contrast the effect of using oracle segmentations and transcriptions in the hybrid system. This is important exercise in order to gain further insights into the effect of these features on system performance. From the previous sections we see that using the oracle segmentations improves the performance of the hybrid system significantly over the baseline HMM system. On the alphadigits task, using the reference segmentations improves the performance of the hybrid system from 11.0% to 7.0% WER (compared to a baseline of 11.9% WER). On the SWITCHBOARD task, the reference segmentation improves the performance of the system from 40.6% to 36.1%

WER (compared to a baseline of 41.6% WER). These numbers do show that the SVMs are capable of locking on to the correct segmentation.

A question that might arise from this experimental setup is — does using reference segmentations eliminate several utterances from the N-best list making the classification task easier for the SVM classifiers? We looked at the average N-best list size after pruning hypotheses that do not satisfy the constraints placed by the reference segmentation in terms of the number of segments. On the SWITCHBOARD task the average list size after pruning was 5.3 as compared the original N-best lists which had an average list size of 8.7. For the alphadigits task this process reduces the average list size from 8.3 to 6.7. It is interesting to note that the performance of the baseline HMM system does not change by using the reduced N-best lists. The reason that the 10-best lists have an average list size of 8.3 is that there are many utterances for which the number of unique word hypotheses in the N-best list is less than 10. Though the list size was reduced using a single segmentation, the SVM classifiers still have to deal with a significant number of confusions and appear to do a better classification job than HMMs.

We ran another set of *oracle* experiments to see the effect of the richness of N-best lists on the performance of the hybrid system. The N-best list error rate was artificially reduced to 0% by adding the reference to the original 10-best lists. Rescoring these new N-best lists using the corresponding segmentations resulting in error rates of 38.1% and 9.1% WER on SWITCHBOARD and alphadigits respectively. This improvement is even more significant in the case of alphadigits where adding the reference transcription to the N-best list improves performance by 30% relative to the baseline. The HMM system

S. No.	Information Source		HMM		Hybrid	
	Transcription	Segmentation	AD	SWB	AD	SWB
1	N-best	Hypothesis	11.9	41.6	11.0	40.6
2	N-best	N-best	12.0	42.3	11.8	42.1
3	N-best + Ref.	Reference	—	—	3.3	5.8
4	N-best + Ref.	N-best + Ref.	11.9	38.6	9.1	38.1

Table 12. Summary of recognition experiments using the baseline HMM system and the hybrid system on the SWITCHBOARD (SWB) and Alphadigits (AD) tasks. The two information sources that define the experimental setup are the transcriptions that need to be reordered and the segmentations that are fed to the hybrid system. N-best segmentation implies that each of the N segmentations were used to process the corresponding hypothesis in the N-best list.

under a similar condition improves performance to 38.6%. On the alphadigits task the HMM system does not improve performance over the baseline even when the reference (or correct) transcription is added to the N-best list. This result indicates that SVMs do a better job than HMMs when they are exposed to accurate segmentations. This issue is addressed in greater detail in a later section.

Another set of experiments were run to quantify the absolute ceiling in performance improvements the SVM hybrid system can provide. This ceiling can be achieved when we use the hybrid system to rescore the N-best lists that include the reference transcription using the reference-based segmentation. Using this setup the system gives a WER of 5.8% on the SWITCHBOARD task and 3.3% on the alphadigits task. This huge improvement should not be mistaken to be a real improvement in performance for two reasons. First, we cannot guarantee that the reference segmentation is

available at all times. Second, generating N-best lists with 0% WER is extremely difficult, if not impossible for conversational speech. This improvement should rather be viewed indicator of the fact that by using good segmentations to rescore good N-best lists, the SVM/HMM hybrid system has a potential to improve performance.

Table 12 summarizes the important results in terms of the various segmentations and N-best lists that were processed to arrive at the final hypothesis. The key point to be noted here is that experiments 2 and 4 are designed such that both the hybrid system and the HMM system are operating under the same conditions and offer a fair comparison of the two systems. For these experiments, since we reorder N-best lists by using segmentations corresponding to each of the hypothesis in the list, both systems have the opportunity to evaluate the same segments. On the other hand if we were to run the experiments using a single segmentation (experiment 1 for example), the HMM system cannot use the segmentation information while the hybrid system can. Experiments 2 and 4 are key in order to compare both systems from a common point of reference. Experiment 4 suggests that when the HMM and hybrid system process good segmentations and rich N-best lists, the hybrid system outperforms the HMM system — significantly in the case of alphadigits and marginally on SWITCHBOARD. This is a very promising result which shows that the hybrid system developed in this dissertation improves performance over the baseline HMM technology.

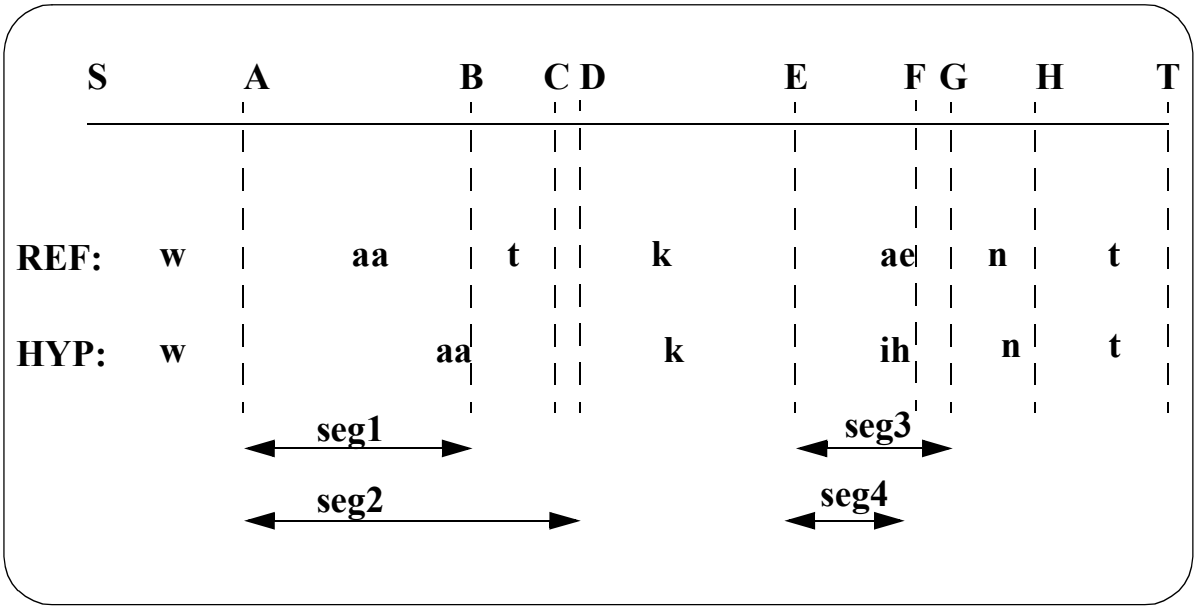

Figure 24. An example of segmentation differences between using the reference transcription and the hypothesis transcription.

7.5. Segmentation Issues in the Hybrid Framework

The effect of reference segmentations on the performance of the hybrid system warrants a closer look at the role segmentation plays in the recognition process. Another aspect that needs to be clarified is the disparity in data that the SVM classifiers are fed during training and testing. It is clear from the numbers presented in sections (7.2.4) and (7.3.2) that reference segmentation (also called the *oracle)* outperforms the hypothesis-based segmentations in all cases. We use a graphic example shown in Fig. 24 to show why we think this is reasonable, if not expected.

The example shows the reference transcription and the hypothesis obtained by the HMM system. SVMs classifiers are used to rescore N-best lists using composite feature vectors generated via the hypothesis-based segmentation. There two types of errors that

we expect the SVM classifiers to handle. Type 1 errors are the errors that occur because of incorrect segmentation (possibly due to insertions or deletions) and are shown as *seg1 vs. seg2.* Type 2 errors are substitution errors (*ae* substituted by *ih*) shown as *seg3* vs *seg4.* Since the SVM classifiers are trained using alignments of reference transcriptions, they learn to handle type 2 errors effectively. However, during the training process, they are not exposed to type 1 errors.

In the hybrid recognition framework, the classifiers do encounter both types of errors, especially when rescoring N-best lists. For tasks such as alphadigits where minimal pairs are the primary error modalities, type 2 errors are dominant and SVMs do a good job of correcting these errors, thereby improving performance compared to the baseline HMM system. However, for SWB, where the baseline error rates are high, type 1 errors are a significant part of the error distribution. Since SVMs trained as described in section 7.2.2 are not intended to learn these type 1 errors, gains are minimal on SWB.

This observation prompts us to hypothesize that subjecting the SVM classifiers to a variety of segmentations during classifier estimation will help provide significant performance improvements on challenging tasks such as SWB. Segmentation variety can be obtained in several ways. One direct method includes generating alternate segmentation graphs for the training data. This would involve generating N-best lists or word-graphs for the training data. Segments that are a *few* frames different from the reference segmentation should be fed as out-of-class data for the classifier being trained on the phone described by the reference segment. Obviously, segments that represent

substituted phones would also be part of the out-of-class data. This is similar in principle to the MMI estimation of HMM parameters described in section 3.1.

7.6. Identification of Mislabeled Data

Issues related to practical SVM optimization were discussed in section 4.4. Choosing a good working set was identified as the key to an efficient training process. Apart from choosing a good working set, the optimization process can be made efficient by identifying support vectors, whose multipliers are at the upper bound, C, early in the training process. In general, if the training errors are not thrown out of the training process, they end up as part of the definition of the hyperplane. In datasets with a high degree of class overlap or large number of mislabelled data points, the training errors can become significant and result in a very complex classifier. The complexity of the classifier effects runtime performance directly, since the number of kernel computations increases linearly with the number of support vectors.

This problem can be handled in several ways [148]. A simplistic approach would be to remove mislabelled data or training errors from the data set before the classifiers are estimated. This is not entirely feasible in most cases because of the cost involved in identifying the mislabeled data. Another approach would entail removing the support vectors whose multipliers are at the upper bound from the definition of the hyperplane. This would reduce the computational complexity significantly, but would result in a hyperplane which is different from the one obtained by the SRM optimization process.

Another serious problem with this approach is that it is blind to the fact that training errors could provide valuable information regarding class overlap.

A third approach to this problem of handling the training errors that become part of the classifier definition is called the Reduced Set method [149]. This attempts to replace the hyperplane definition by a new definition using a function estimation technique. The new definition would be formulated in terms of a much smaller support vector set which is composed of vectors that may not be a part of the training data. This technique is elegant in that it guarantees that the new hyperplane is very close to the original hyperplane under a pre-defined error metric. However, the estimation process for the new hyperplane is very expensive and a closed form solution exists only for 2^{nd} order polynomial kernels.

A fourth technique used as part of the estimation process of the classifiers used in this dissertation involves an iterative procedure where training errors are removed from the support vector set. This is followed by retraining the hyperplane with the new reduced set as the base support vector set. This approach is a compromise in terms of computational cost and reduced classifier complexity. A by-product of this approach is the ability to efficiently identify mislabeled data as part of the training process. The key to the success of this approach in identifying mislabeled data is the use of large value for C during classifier estimation. Using a small value for C could result in vectors being incorrectly identified as training errors or mislabeled data. This can have detrimental effect on the classifier's performance.

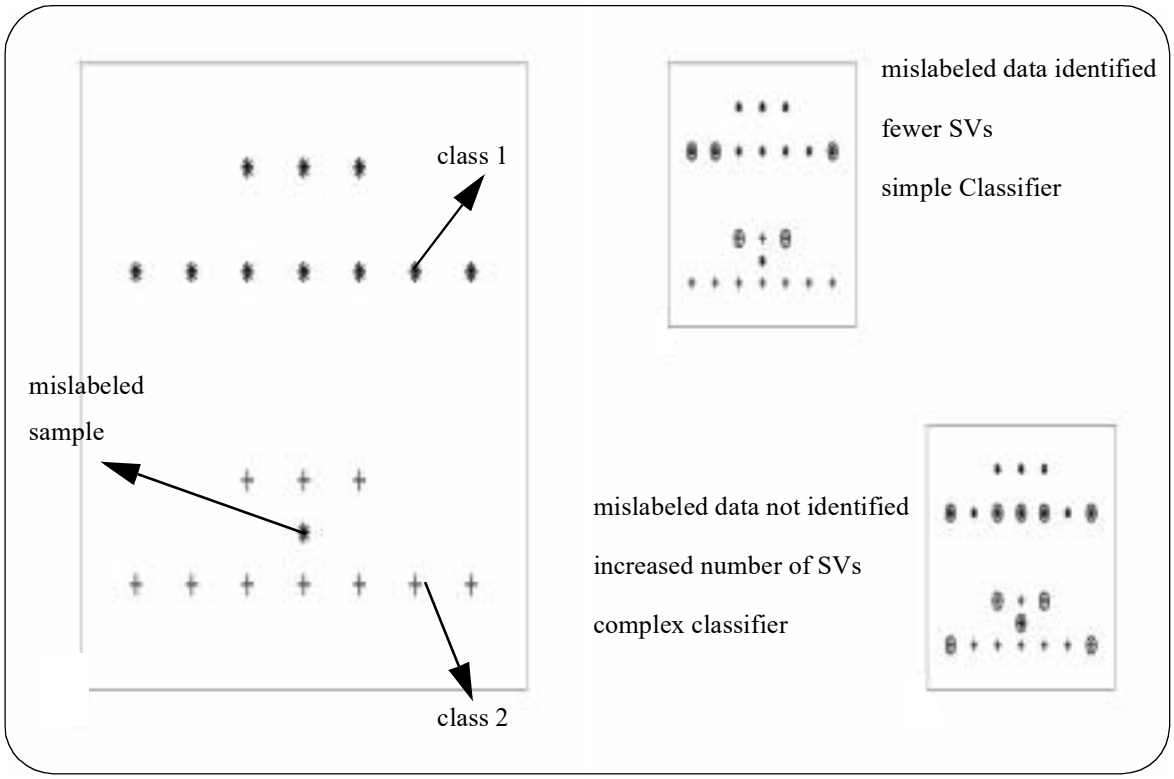

Figure 25. A two-class dataset in which one sample belonging to class 1 is intentionally labeled as belonging to class 2. The figure on the right compares the support vectors when the mislabeled sample is identified to the case when it is not correctly identified.

Analysis of several experiments on noisy data (for example, classes with significant overlap possibly due to mislabeling) revealed that there are often several support vectors with their multipliers at the upper bound C. When an example has its multiplier consistently at the upper bound across iterations of the chunking algorithm used to train classifiers, it is a good indication that the example is either an outlier, an area of overlap between features, or is mislabeled data. Removing these bounded support vectors from the optimization problem speed-up later iterations by reducing the size of the sub-problem that needs to be solved (ref. section 4.4). Note that when the bounded support vectors are not removed from the optimization process, especially when C is large, they

C	mislabeled data not identified		mislabeled data identified	
	# of SVs	% error	# of SVs	%error
1	3644	1.9	1747	3.6
5	3833	1.8	3329	1.8
10	3854	1.7	3715	1.7
20	3854	1.7	3745	1.8
50	3854	1.8	3854	1.8

Table 13. Effect of the parameter C on the performance improvements obtained by eliminating mislabeled data from the classifier estimation process.

end up as support vectors for the final solution. In the case of noisy data this could lead to inaccurate decision surfaces and bad generalization.

Fig. 25 shows a synthetic data example where identifying the bounded support vector greatly simplifies the classifier. The effect of C on this process can be seen in Table 13. This table shows the performance of a classifier for the phone *b* of the OGI alphadigits dataset using an RBF kernel with $gamma = 0.6$. For $C = 5$, we see that the number of support vectors can be reduced by 16% without a significant effect on the classifier accuracy. However, using a small value for C,such as $C = 1$, can significantly degrade the classifier's performance. This trend is common to all classifiers and more significant for classes with large data overlap. The cross-validation set was used to perform this analysis.

We attempted to calibrate the effect of this property on the performance of the hybrid system. We trained classifiers using the same dataset partitioning as described in the previous section. We trained two sets of classifiers. The first set included classifiers which allow mislabeled data to be a part of the hyperplane definition. The other set included classifiers where mislabeled data points are identified and not allowed to become part of the support vector set. Our goal was to compare the effect of identifying mislabeled data on the classifier accuracy and classifier size. The RBF kernel was used with $gamma = 0.6$ and $C = 5$. Identifying mislabeled data resulted in classifiers that had 22% fewer support vectors on an average. These classifiers when used as part of the hybrid system resulted in a 12.1% WER. This can be compared to a 13.5% WER for a system in which the mislabeled data was not identified. The performance of these systems is in general worse than our best reported WER (11% — see Table 8) for the hybrid system. This is due to the fact that a smaller value for $C = 10$ was used for the estimation of the classifiers to better illustrate the effect of data cleanup as well as to keep the classifiers small. At $C = 50$, very few examples are identified as outliers which makes the classifiers very complex.

7.7. Summary

This chapter presents the performance of SVM classifiers and the hybrid SVM/ HMM system in classifying static and dynamic speech patterns. The preliminary experiments on the Deterding vowel task demonstrated that SVMs were competitive with other classification schemes. This provided motivation to explore the use of SVMs as the

core of a speech recognition system. We used a hybrid framework for this purpose. In order to gauge the performance of the hybrid system, the OGI alphadigits and SWITCHBOARD tasks were used. On the alphadigits task, the hybrid system obtains a WER of 10.6% compared to 11.9% obtained using the baseline HMM system The hybrid system clearly improves performance on all word classes including minimal pairs. On the SWITCHBOARD task, the improvements were less dramatic, 40.6% WER compared to a baseline of 41.6%. In this system the SVM classifiers were used to rescore 10-best lists which had a list error rate of 29.5%.

We conjecture that the variety in segmentation constrains the improvements achieved by the hybrid system. In the oracle experiments we added the reference transcription to the N-best lists to see if the richness of N-best lists has an effect on the system performance. In these experiments we see that the SVM classifiers seem to lock on to the correct hypothesis better than HMM models when good segmentations are used. These encouraging results show that SVMs are indeed doing a better job at classification but the segmentation issue needs to be addressed.

We have also explored the use of SVMs to identify mislabeled data. This is a very useful by-product of the classifier estimation process which can have a serious impact on training classifiers using imperfect transcriptions. This property of identifying mislabeled data or outliers can be extremely useful in processing large datasets which have significant transcription errors [150]. The process can also be extended to provide confidence measures [151].

CHAPTER 8

CONCLUSIONS AND FUTURE DIRECTIONS

This dissertation addresses the application of Support Vector Machines (SVM) to the continuous speech recognition problem. The technology has been applied to a small vocabulary task — OGI Alphadigits, and a large vocabulary conversational speech task — SWITCHBOARD. The results obtained clearly validate the classification power of SVMs and support the use of SVMs for acoustic modeling. A significant contribution of this dissertation is a hybrid SVM/HMM system which uses SVMs to post process data generated by a conventional HMM system. This framework has been proven to provide significant improvements in recognition accuracy on OGI Alphadigits and marginal improvements on SWITCHBOARD. The application of SVMs to various other aspects of speech recognition, including phone classification and the identification of mislabeled data have been introduced.

8.1. Support Vector Machine Classifiers

Most speech recognition systems today are based on hidden Markov models (HMM) and a few are based on hybrid HMM-Neural Network architectures. HMMs have had significant success since they offer an elegant mechanism to model both the acoustic variability and the temporal evolution of speech. The existence of efficient

iterative parameter estimation procedures like the Expectation-Maximization (EM) algorithm has a significant role in the universal usage of HMMs in speech recognition systems. However, HMMs suffer from a number of drawbacks — the assumption of independence of successive frames and the standard maximum likelihood (ML) approach to HMM parameter estimation being the foremost.

The assumption of independence has been countered to a certain extent by augmenting the feature vectors with the delta and acceleration components. Neural networks are better at addressing this issue with the use of feedback as in a recurrent neural network or the use of multi-frame data. SVM classifiers can also handle multi-frame data efficiently because of their ability to handle high-dimensional inputs. Using multi-frame data allows the classifier to learn the correlations directly from the data instead of making independence assumptions. In this dissertation, though multi-frame data has not been used explicitly, a variation of the same idea which uses segment averages has been applied successfully.

SVMs overcome some of the limitations of neural networks because of the underlying criterion for parameter estimation, Structural Risk Minimization (SRM). SRM allows for parameter estimation where the generalization ability of the system can be controlled as part of the learning process. SVMs have been applied successfully to several classification tasks where they have out performed neural networks. The results reported in this dissertation also confirm this fact. SVMs have the ability to handle high-dimensional data effectively where the important dimensions are learned automatically. Their learning process is inherently discriminative. The problems inherent

in HMM systems and the limitations of neural networks have motivated us to explore the use of SVMs for speech recognition.

8.2. Dissertation Contributions

SVMs have been applied to speech recognition via a set of experiments ranging from simple vowel classification to complex conversational speech tasks. Several important issues have been addressed in the process — some common to hybrid systems and some specific to the use of SVMs with speech data. These issues are summarized in the following sections.

8.2.1. Hybrid Recognition Architecture

As a first step towards using SVMs as an alternate approach to speech recognition we explored a hybrid framework where SVMs are used as part of a post-processing stage. This hybrid approach uses SVMs to process information supplied by a baseline HMM system to arrive at the final hypothesis. The baseline HMM system is used to provide segmentations in order to construct the input feature vectors for the SVMs. The HMM system also provides N-best lists that are rescored by the SVM classifiers.

This approach to using SVMs for speech recognition is different from other attempts to incorporate SVMs into a speech recognition system. Our system has a distinct two stage process unlike systems described in [152, 153] where HMMs and SVMs are tied together more closely via Fisher Kernels. This concept is described in detail in a later section of this dissertation.

8.2.2. SVM Distance to Likelihood Mapping

SVMs are commonly used as binary classifiers where the classifier learns the decision region that separates the in-class and the out-of-class data. Neural networks are also capable of learning such a decision region, but have one additional advantage. It has been shown that neural network outputs model the posterior probability $P(class/data)$. Such a model is crucial to integration of this technology into an HMM framework. As shown in (127), SVMs classify data based on distances. In order to integrate SVMs into an HMM framework, we need to convert these distances to posterior probabilities. Several schemes have been studied to convert SVM distances to likelihoods in order to fit the SVM classifiers into the HMM-based ASR system. The sigmoid-based warping function has been found to be sufficiently accurate and has been used in all experiments presented in this dissertation.

8.2.3. Segment Level Data

One of the primary motivations for the use of SVMs in speech recognition is the discriminative learning paradigm under which the classifiers are estimated. However, just like other discriminative techniques, SVM classifier estimation can be very slow when several hundred thousand training samples are involved in the estimation process. This is definitely the case with speech when classification is done at the frame level. For example, a training set of 10 hours of speech data is composed of $36x10^5$ training samples. In a situation where we train binary classifiers, each classifier learns from all the $36x10^5$

samples. This could take a prohibitive amount of CPU time for estimation considering the fact that we estimate about 40 classifiers in a typical speech application.

In order to make SVM training feasible for a speech application, a simple solution is to learn to classify at a coarser level of granularity than the frame-level. This approach, while reducing the computational complexity, has other advantages from a hybrid recognition system's perspective. Since the hybrid system involves bootstrapping SVM classifiers using a traditional HMM system, classification at a coarser level takes a small fraction of the total recognition time. The additional overhead for the SVM portion of the recognition process is minimal.

Segment level data was used for all recognition experiments, i.e. SVM classifiers were trained to classify segments of speech instead of frames of speech. The unit of recognition typically used in experiments reported here was a phone. Each phone is assumed to be composed of three segments, one each for the onset, nucleus, and coda. A composite vector comprised of the averages for each of the segments is created by simple concatenation. Thus, every phone is represented by a composite feature vector which is used for classification. Several combinations of segment ratios have been experimented with and the differences in performance are insignificant.

8.2.4. N-Best List Generation

The SVM classifiers are trained using both in-class and out-of-class data. In theory, if the feature space in which the classifiers operate is chosen judiciously, classes that are phonetically similar are closer to each other as compared to the classes that are

well separated in the phonetic space. With this assumption, which has been corroborated by the phone confusion matrices generated from classification data, it is hypothesized that the power of the SVM classifiers can be better used by operating on N-best lists which provide a compact representation of the acoustic confusion space as seen by the traditional HMMs.

The SVM classifiers in the hybrid recognition system operate on N-best lists generated using the baseline HMM system. As part of this dissertation, N-best list generation has been added to the ISIP ASR Toolkit, using an A*-search based implementation.

8.2.5. Identification of Mislabeled Data

One of the key aspects of classifier design is the ability to add information to the data that allows the classifier to focus its modeling power on the decision surface that separates the class from most samples of the out-of-class data. Any typical hand-labeled or automatically-labeled database has an inherent labeling error. In some cases this can be significant. For example, the original version of SWITCHBOARD Corpus had a label WER of ~8% [150]. These labeling errors can force the classifiers to expend significant resources in modeling the incorrect labels at the cost of diminished generalization.

We have developed a scheme in the SVM estimation process where the mislabeled data can be identified accurately during classifier estimation. The concept behind this approach is that outliers and mislabeled data typically violate the constraints in the SVM optimization functional and do so consistently over several iterations of the optimization

process. Such training samples end up as bounded support vectors. In noisy data with significant amounts of mislabeling, we eliminate the bounded support vectors from the optimization process. This approach has been tested on both synthetic and real data and its effectiveness has been proven in terms of estimating compact classifiers with improved generalization.

8.3. Summary of Experiments

8.3.1. Static Classification Tasks

SVMs have been evaluated on a standard non-linear classification task, Deterding Vowel data. This task is challenging because of the inherent data overlap and a sparse training set. SVM classifiers achieved a classification error rate of 35% which is better the performance reported using several other non-linear classification techniques including neural networks. This dataset has also been used to study the effect of kernel parameters on classifier accuracy. RBF kernels consistently outperformed polynomial kernels.

8.3.2. Speech Recognition

We evaluated the hybrid SVM/HMM speech recognition architecture on two continuous speech recognition tasks. The OGI Alphadigits dataset was used as a small vocabulary task and the SWITCHBOARD task was used as the large vocabulary task to calibrate the improvements achievable using the hybrid system.

The baseline HMM system for the alphadigits task used context-dependent phone models with 12 Gaussian mixture components per state. This configuration achieved a

WER of 11.9% on the speaker independent evaluation set. In comparison, rescoring 10-best lists with SVM classifiers using RBF kernels, we achieved a WER of 11.0%. Alignments based on the output of the baseline HMM system were used for this purpose. The system that used a score combination mechanism, where the likelihoods from the HMMs were combined with the likelihoods of the SVM classifiers, achieved a WER of 10.6%, which is a 9% relative improvement over the baseline.

For the SWITCHBOARD task we evaluated the hybrid system using SVM classifiers with RBF kernels only. The baseline HMM system for this task was a context-dependent phone system with 12 Gaussian mixtures per state. This system achieved a WER of 41.6% on a development test set consisting of 2,427 utterances. The hybrid system achieved a WER of 40.6% when RBF kernels with $gamma = 0.1$ were used. 10-best lists with a list WER of 29.5% were used as input to the hybrid system. The segmentations for the hybrid system were based on the baseline HMM systems' best hypothesis.

8.3.3. Analysis and Oracle Experiments

In order to study the effect of segmentation on the performance of the hybrid system, we ran several experiments on the both speech recognition tasks using various segmentation schemes. In the first experiment, we compared the performance of a system that used N segmentations to reorder an N-best list to a system that reordered an N-best list using a single segmentation. The former setup gave worse performance suggesting that

SVMs were having problems when a variety of segmentations are used — 11.0% vs. 11.8% WER on the alphadigits task and 40.6% vs. 41.6% WER on SWITCHBOARD.

In another set of experiments, we augmented the N-best lists with the reference transcription thereby reducing the list error rate to 0%. Under this scheme, the hybrid system improved performance significantly indicating that the SVMs seem to lock on to the correct segmentation — 9.1% vs. 11.8% baseline for alphadigits and 38.1% vs. 41.6% baseline for SWITCHBOARD. As a point of comparison the HMM system was evaluated using the same augmented N-best list. This system did not improve performance as much as the hybrid system — 11.9% vs. 11.9% baseline for alphadigits and 38.6% vs. 41.6% baseline for SWITCHBOARD.

The above experiments clearly suggest that the SVM classifiers do a good job when good segmentations are available. We hypothesize that providing the SVM classifiers with a variety of segmentations when estimating parameters will decrease the extent of this dependence on good segmentations. This issue has been discussed in section 7.3.2.

8.4. Future Work

Though we have seen that SVMs provide consistent improvement over HMM systems, there are several aspects of the hybrid system that need to be researched further. Some of the topics that need to be addressed further relate to building better SVM classifiers, while the others relate to the hybrid system architecture.

8.4.1. Posterior Estimation

In this work, we have used a sigmoid function to approximate the posterior probability distribution of the classes that the SVMs are used to model. However this is not the best approximation of the posterior. Also, this approach requires the need to have a cross-validation set, which can be expensive in some cases. We need to explore approaches where the posteriors can be estimated without the need for cross-validation sets. Also, the sigmoid approximation does not account for the inherent overlap in the data. New approaches such as Relevance Vector Machines which are grounded in probability theory have a clear advantage over SVMs [126] in this regard.

Apart from the issue of posterior estimation, we need to explore ways of simultaneously optimizing the sigmoids such that the sum of the posteriors across all classifiers is a true probability distribution (i.e. sums to unity). This will improve the discrimination capability of the classifiers. One approach to achieve this is to train a neural network to estimate the posterior constraining the output nodes to sum to one.

8.4.2. Data Cleanup and Confidence Measures

In section 7.6 we have seen the effect of identifying mislabeled data on classifier estimation. This ability of the SVM estimation process to identify mislabeled data can be used for other applications such as data cleanup. Another interesting avenue for future work is to develop a mechanism to assign a confidence measure to this process where data points that are identified as mislabeled data are done so with a confidence rating. The mislabeled data can then be appropriately post processed based on the confidence scores.

8.4.3. SVM Parameter Update

In section 4.4 we discussed the SVM parameter estimation process. However, this process is based on a fixed training set. In a hybrid architecture, we can estimate better classifiers if we had a mechanism to iteratively recognize the training data and appropriately feed the errors back into the next iteration of parameter estimation. This form of iterative estimation has proven to be very valuable for neural network based hybrid systems [51].

8.4.4. Effect of Segmentation

In section 7.5, we have demonstrated why segmentation is an important issue for the hybrid system developed in this dissertation. The performance of the hybrid system in the *oracle* experiments show that segmentations dramatically affect performance. This problem can be addressed in two phases. Similar to MMI estimation of HMM parameters, during training we can use a word graph (which provides alternate hypothesis and segmentations) to estimate the SVM classifiers. In other words, for every positive example in a classifier, the training data should include the competing segments centered around the segment corresponding to the example. During testing, instead of using a single segmentation to generate the acoustic data for the SVM classifiers, a segment graph can be rescored.

8.4.5. Alternate Hybrid Approaches

In this dissertation we used a hybrid approach where the SVMs post-processed information provided by an HMM-based system. However the information was only used primarily to compute the input feature vectors for the SVM classifiers. A different hybrid approach could attempt to more closely integrate the HMMs and SVMs by using SVM classifiers to discriminatively learn the state sequence characteristics of the HMM system. The SVMs can also be used as function estimators to learn the emission probabilities of the states in an HMM. This approach has been addressed in preliminary experiments using transition-based HMMs [154]. The next section describes a similar approach which ties in HMM parameters to SVMs more closely.

8.4.6. Kernels and Sufficient Statistics

SVM distances provide a measure that can be used to compare any two input vectors directly for classification purposes. Traditional HMMs on the other hand use log-likelihood as a metric to compare vectors. Log-likelihood only provides a measure of closeness of the vector to the model itself and cannot be effectively used to measure distances between vectors or to compare vectors directly for classification purposes. In other words in order to compare two vectors using Gaussians as classifiers in an HMM framework, we need to indirectly compare the two via their closeness to the model. It may also happen in many cases that two different Gaussians may result in the same likelihood for two completely different input vectors. This can happen because each of the Gaussian classifiers was estimated in isolation of the other.

In this section we introduce the theory for building an SVM within an HMM framework in order to allow for direct comparison of input vectors while allowing for effective modeling of dynamic patterns. This mechanism will allow us to use the representative power of Gaussian-based HMMs (as well as their ability to handle temporal variations) in conjunction with the discriminative power of SVMs [153].

The vector of sufficient statistics describes the process of generating the underlying state sequence from the HMM. For example, the quantities in this vector would include the posterior probabilities of taking a particular state transition or emitting a particular observation vector for a state. In traditional HMM parameter estimation, the log likelihood of the data given the model is the objective function. Thus, any observation sequence can be converted to a fixed length feature vector comprised of the sufficient statistics. Instead of dealing with the sufficient statistics directly, we can work with the gradients described in the appendix to this dissertation. These gradients are also called the Fisher Scores [155]. Fisher scores are part of the commonly used definition for the Fisher Information Matrix [124,156]. The Fisher matrix is often viewed as the information contained in the data set about each parameter.

We have seen that maximum likelihood based HMM parameter estimation involves the maximization of $P(O/M)$ where M is the parameter set that defines the model and O is the input acoustic evidence. In order to describe the training data, all algorithms based on stochastic optimization will need the gradient of an objective function with respect to the parameters of the HMM. It is the solution to these gradients that gives

us the optimal estimated value. The Fisher information matrix I, for this case is defined as,

$$I = E_O\left\{ U_O U_O^T \right\}, \text{ where} \tag{138}$$

$$U_O = \nabla_M \log P(O/M) \tag{139}$$

and the expectation is over $P(O/M)$. U_O is called the Fisher score. We can define a distance metric for two observation sequences O_i and O_j mapped to this model M as,

$$K(O_i, O_j) = U_{O_i} I^{-1} U_{O_j} \tag{140}$$

in terms of the Fisher scores. The right-hand side of the above equation is positive definite and hence can be a valid kernel as per Mercer's conditions [28]. Note how the above definition of a kernel ties in our notion of HMM parameter estimation with kernels that form an integral part of SVM theory. The Fisher kernel defined above can be used instead of other kernels defined in the previous chapter to determine a separating hyperplane in the Fisher score space or a higher dimensional space. This method will theoretically perform at least as well as the underlying HMM model [155]. For practical reasons, the Fisher information matrix is often chosen as an identity matrix [155]. This assumption, though strong, can be justified if the Fisher scores are pre-whitened. We now have a simple procedure where the advantages of the HMM model are available to the discriminative classifier, SVM, via the Fisher kernel. With logistic regression models, I^{-1}

can be viewed as the covariance matrix of a Gaussian prior and is often discarded from (140). We do not however know if that is true in the case of traditional HMMs. In the appendix the mathematical derivation to compute the Fisher scores is provided.

8.5. Summary

In this chapter, we summarized the advantages of using SVMs as acoustic models in a speech recognition system. The experiments conducted as part of this work clearly indicate that SVMs are a promising option for acoustic modeling especially in a hybrid framework. The work reported in this dissertation is the first successful attempt at integrating SVMs into a complex speech recognition system. SVMs provide unique advantages along several dimensions in solving the speech recognition problem. The effect of SVMs being a discriminative classification scheme transforms to significant performance improvements in classification at the phone-level and overall recognition accuracy at the word-level. The *oracle* experiments show great promise for this new technology if we address the segmentation issue appropriately. Several avenues of further research have been suggested that include approaches to improve the SVM classifier estimation and the hybrid framework.

REFERENCES

[1] S. Harnad, *Categorical Perception: The Groundwork of Cognition*, Cambridge University Press, New York, 1997.

[2] A. Martin and M. Przybocki, "Analysis of Results - The 2001 NIST Hub-5 Evaluation," *http://www.nist.gov/speech/tests/ctr/h5_2001/postwshp.htm*, May 2001.

[3] W.J. Ebel and J. Picone, "Human Speech Recognition Performance on the 1994 CSR Spoke 10 Corpus," *Proceedings of the Spoken Language Systems Technology Workshop*, pp. 53-59, Austin, Texas, USA, January 1995.

[4] N. Deshmukh, A. Ganapathiraju, R.J. Duncan and J. Picone, "Human Speech Recognition Performance on the 1995 CSR Hub-3 Corpus," *Proceedings of the Speech Recognition Workshop*, pp. 129-134, Harriman, NY, USA, February 1996.

[5] A. Acero. *Acoustical and Environmental Robustness in Automatic Speech Recognition*, Ph. D. dissertation, Carnegie Mellon University, Pittsburgh, USA, 1990.

[6] Pallett, D., et. al., "1995 Benchmark Tests for the ARPA Spoken Language Program," in *Proceedings of the SLST Workshop,* Harriman NY, February 1996.

[7] Intel Developer Services, "An Overview of Speech Technology and Applications," *http://developer.intel.com/software/idap/resources/technical_collateral/pentiumii/speech.htm*, September 1998.

[8] F. Kubala, et. al., "Rough'n'Ready: a meeting recorder and browser," *ACM Computing Surveys*, vol. 31, issue 2, 1999.

[9] H.D. Wactlar, et. al., "Informedia Experience-on-Demand: capturing, integrating and communicating experiences across people, time and space", *ACM Computing Surveys*, vol. 31, issue 2, 1999.

[10] J.W. Butzberger, et. al., "Spontaneous speech effects in large vocabulary speech recognition applications," *Proceedings of DARPA Speech and Natural Language Workshop*, Morgan Kaufmann, pp. 339-343, 1992.

[11] E.E. Shriberg, *Preliminaries to a Theory of Speech Disfluencies*, Ph. D. dissertation, University of California, Berkeley, USA, 1994.

[12] A. Stolcke and E. Shriberg, "Statistical language modeling for speech disfluencies," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, pp. 405-408, Atlanta, GA, May 1996.

[13] E. Shriberg, "Disfluencies in SWITCHBOARD," *Proceedings of the International Conference on Spoken Language Processing*, Vol. Addendum, pp. 11-14, Philadelphia, PA, October 1996.

[14] E. Shriberg, "Phonetic Consequences of Speech Disfluency," *Symposium on The Phonetics of Spontaneous Speech — Proceedings of the International Congress of Phonetic Sciences,* Vol. 1, pp. 619-622, San Francisco, CA, 1999.

[15] J. Godfrey, E. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone Speech Corpus for Research and Development," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 517-520, San Francisco, California, USA, March 1992.

[16] J. Picone, "Signal Modeling Techniques in Speech Recognition," *IEEE Proceedings*, vol. 81, no. 9, pp. 1215-1247, September 1993.

[17] S. Furui, "Cepstral Analysis Technique for Automatic Speaker Verification," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 29, No. 2, pp. 254-272, April 1981.

[18] N. Kumar, and A. Andreou, "Generalization of linear discriminant analysis within the maximum likelihood framework," *Proceedings of the Joint Statistical Meeting of the American Statistics Association*, Chicago, IL, August 1996.

[19] V. Valtchev, *Discriminative Methods in HMM-based Speech Recognition*, Ph. D. dissertation, University of Cambridge, UK, 1995.

[20] Y. Normandin, *Hidden Markov Models, Maximum Mutual Information Estimation, and the Speech Recognition Problem*, Ph. D. dissertation, McGill University, Canada, 1991

[21] B.-H, Juang and S. Katagiri, "Discriminative Training," *Journal of the Acoustical Society of Japan*, vol. 13, pp. 1404-1413, 1992.

[22] P.C. Woodland and D.R. Cole, "Optimizing Hidden Markov Models using Discriminative Output Distributions," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, April 1991.

[23] B.-H. Juang and S. Katagiri, "Discriminative Learning for Minimum Error Classification," *IEEE Transactions on Signal Processing*, vol. 40, no. 12, pp. 3043-3054, 1992.

[24] C. Cortes, V. Vapnik. Support Vector Networks, *Machine Learning*, vol. 20, pp. 293-297, 1995.

[25] C.J.C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, http://svm.research.bell-labs.com/SVMdoc.html, AT&T Bell Labs, November 1999.

[26] E. Osuna, R. Freund, and F. Girosi, "Support Vector Machines: Training and Applications," *MIT AI Memo 1602*, March, 1997.

[27] B. Schölkopf, *Support Vector Learning*, Ph.D. dissertation, R. Oldenbourg Verlag Publications, Munich, Germany, 1997.

[28] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, NY, USA,1995.

[29] B. Schölkopf, C. Burges and A. Smola, *Advances in Kernel Methods: Support Vector Machines*, MIT Press, Cambridge, MA, USA, December 1998.

[30] M.A. Hearst, et. al., "Trends and Controversies - Support Vector Machines", *IEEE Intelligent Systems*, vol. 13, pp. 18-28, 1998.

[31] E. McDermott, *Discriminative Training for Speech Recognition*, Ph. D. dissertation, Waseda University, Japan, 1997.

[32] P. Woodland and D. Povey, "Very Large Scale MMIE Training for Conversational Telephone Speech Recognition," *Proceedings of the 2000 Speech Transcription Workshop,* University of Maryland, MD, USA, May 2000.

[33] D. Pallett, et. al., "Overview: Speech Transcription Workshop," *Proceedings of the 2000 Speech Transcription Workshop*, University of Maryland, MD, USA, May 2000.

[34] R. Rosenfeld, *Adaptive Statistical Language Modeling: a Maximum Entropy Approach*, Ph. D. dissertation, Carnegie Mellon University, Pittsburgh, USA, 1994.

[35] P. Clarkson and R. Rosenfeld, "Statistical Language modelling using CMU-Cambridge Toolkit," *Proceedings of Eurospeech*, pp. 2707-2710, Rhodes, Greece, September 1997.

[36] J.R. Deller, J.G. Proakis and J.H.L. Hansen, *Discrete-Time Processing of Speech Signals*, Macmillan Publishing, New York, USA, 1993.

[37] J. Picone, "Continuous Speech Recognition Using Hidden Markov Models," *IEEE Acoustics, Speech, and Signal Processing Magazine*, vol. 7, no. 3, pp. 26-41, July 1990.

[38] L.R. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, New Jersey, USA, 1993.

[39] F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Cambridge, Massachusetts, USA, 1997.

[40] J.L. Gauvain, et. al., "The LIMSI 1995 Hub3 System," *Proceedings of the DARPA Speech Recognition Workshop*, pp. 105-111, Harriman, NY, USA, February 1996.

[41] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, San Diego, CA, USA, 1990.

[42] A.P. Dempster, N.M. Laird and D.B. Rubin, "Maximum Likelihood Estimation from Incomplete Data," *Journal of the Royal Statistical Society,* vol. 39, no. 1, pp. 1-38, 1977.

[43] G. McLachlan, *The EM algorithm and extensions*, John Wiley, New York, NY,1997.

[44] R.A. Redner and H.F. Walker, "Mixture Densities, Maximum Likelihood and the EM Algorithm," *SIAM Review,* vol. 26, no. 2, pp. 195-239, 1984.

[45] J.S. Bridle and L. Dodd, "An Alphanet Approach to Optimizing Input Transformations for Continuous Speech Recognition," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 277-280, May 1991.

[46] J.S. Bridle, Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationship to Statistical Pattern Recognition, *Neuro-Computing: algorithms, architectures and applications*, Springer-Verlag, 1989.

[47] M.D. Richard and R.P. Lippmann, "Neural Network Classifiers Estimate Bayesian a Posteriori Probabilities", *Neural Computation*, vol. 3, no. 4, pp. 461-483, 1991.

[48] S. Renals, *Speech and Neural Network Dynamics*, Ph. D. dissertation, University of Edinburgh, UK, 1990.

[49] N. Ström, "Phoneme Probability Estimation with Dynamic Sparsely Connected Artificial Neural Networks," *The Free Speech Journal*, vol. 1, no. 5, 1997.

[50] A. J. Robinson, *Dynamic Error Propagation Networks*, Ph.D. dissertation, Cambridge University, UK, February 1989.

[51] H.A. Bourlard and N. Morgan, *Connectionist Speech Recognition — A Hybrid Approach*, Kluwer Academic Publishers, Boston, USA, 1994.

[52] G.D. Cook and A.J. Robinson, "The 1997 ABBOT System for the Transcription of Broadcast News," *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, VA, USA, 1998.

[53] J. Tebelskis, *Speech Recognition using Neural Networks*, Ph. D. dissertation, Carnegie Mellon University, Pittsburgh, USA, 1995.

[54] M.J. Russell and R.K., Moore, "Explicit Modeling of State Occupancy in Hidden Markov Models for Automatic Speech Recognition," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 5-8, Tampa, USA, 1985.

[55] W. Holmes, *Modelling Segmental Variability for Automatic Speech Recognition*, Ph. D. dissertation, University of London, UK, 1997.

[56] M.J.Russell and W.J. Holmes, "Linear Trajectory Segmental Models," *IEEE Signal Processing Letters*, vol. 4, no. 3, pp. 72-74, 1997.

[57] M. Ostendorf, et. al., "From HMM's to Segment Models: A Unified View of Stochastic Modeling for Speech Recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 5, pp. 360-378, 1996.

[58] M. Ostendorf and S. Roukos, "A Stochastic Segment Model for Phoneme-Based Continuous Speech Recognition," *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 37, pp. 1857-1867, December 1989.

[59] S. Austin, et. al., "Speech Recognition Using Segmental Neural Nets," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 625-628, 1992.

[60] F. Rosenblatt, "The Perceptron: A Perceiving and Recognizing Automaton", C*ornell Aeronautical Laboratory Report 85-460-1*, 1957.

[61] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning Internal Representation by Error Propagation," D.E. Rumelhart and J.L. McClelland (Editors), *Parallel Distributed Processing: Explorations in The Microstructures of Cognition, Vol. 1: Foundations*, MIT Press, Cambridge, MA, pp. 318-362, 1986.

[62] D.A. Ackley, G.E. Hinton and T.J. Sejnowski, "A Learning Algorithm for Boltzmann Machines," *Cognitive Science*, Vol. 9, pp. 147-169, 1985.

[63] U. Bodenhausen, S. Manke, and A. Waibel, "Connectionist Architectural Learning for High Performance Character and Speech Recognition," *Proceedings of the International Conference on Acoustics Speech and Signal Processing,* vol. 1, pp. 625-628, Minneapolis, MN, USA, April 1993.

[64] S. Kirkpatrick, C.D. Gellatt and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, pp. 671-680, 1983.

[65] J. Fritsch, *Hierarchical Connectionist Acoustic Modeling for Domain-Adaptive Large Vocabulary Speech Recognition*, Ph. D. dissertation, University of Karlsruhe, Germany, 2000.

[66] S. Lawrence, C. L. Giles, and A. C. Tsoi, "What size neural network gives optimal generalization," *Technical Report UMIACSTR-96-22*, Institute for Advanced Computer Studies, University of Maryland, USA, April 1996.

[67] S. Lawrence, C.L. Giles, and A.C. Tsoi, "Lessons in neural-network training: Overfitting may be harder than expected," *Proceedings of the 14th National Conference on Artificial Intelligence*, AAAI Press, pp. 540-545, 1997.

[68] V.N. Vapnik, *Statistical Learning Theory*, John Wiley, New York, NY, USA, 1998.

[69] Y. LeCun, et. al., "Handwritten Digit Recognition with Backpropagation Network," *Advances in Neural Information Processing Systems-2*, Morgan Kaufman, pp. 396-404, 1990.

[70] T. Joachims, *SVMLight: Support Vector Machine*, http://-ai.informatik.uni-dortmund.de/FORSCHUNG/VERFAHREN/SVM_LIGHT/svm_light.eng.html, University of Dortmund, November 1999.

[71] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," *Technical Report 23, LS VIII, University of Dortmund*, Germany, 1997.

[72] M. Schmidt, H. Gish, "Speaker Identification Via Support Vector Classifiers," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 105-108, Atlanta, GA, USA, May 1996

[73] S. Fine, J. Navratil and R. A. Gopinath. Hybrid GMM/SVM Approach to Speaker Identification, *Proceedings of the International Conference on Acoustics, Speech and Signal Processing,* Salt Lake City, Utah, USA, 2001.

[74] A. Ganapathiraju, J. Hamaker and J. Picone, "Support Vector Machines for Speech Recognition," *Proceedings of the International Conference on Spoken Language Processing*, pp. 2923-2926, Sydney, Australia, November 1998.

[75] C. Philip and P. Moreno, "On the Use of Support Vector Machines for Phonetic Classification," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Phoenix, Arizona, USA, 1999.

[76] A. Ganapathiraju, J. Hamaker and J. Picone, "A Hybrid ASR System Using Support Vector Machines," *Proceedings of the International Conference of Spoken Language Processing*, vol. 4, pp. 504-507, Beijing, China, October 2000.

[77] A. Ganapathiraju, J. Hamaker and J. Picone, "Hybrid HMM/SVM Architectures for Speech Recognition," *Proceedings of the Department of Defense Hub 5 Workshop*, College Park, Maryland, USA, May 2000.

[78] N. Deshmukh, A. Ganapathiraju and J. Picone, "Hierarchical Search for Large Vocabulary Conversational Speech Recognition," *IEEE Signal Processing Magazine*, vol. 1, no. 5, pp. 84-107, September 1999.

[79] N. Deshmukh, et. al., "A Public Domain Speech-to-Text System," *Proceedings of. Eurospeech,* vol. 5, Budapest, Hungary, September 1999.

[80] M. Berouti, et. al., "Enhancement of Speech Corrupted by Acoustic Noise," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 208-211, 1979.

[81] X. Huang, et. al., *Spoken Language Processing*, Prentice Hall, Englewood Cliffs, New Jersey, USA, 2001.

[82] A. Ganapathiraju, V. Goel, J. Picone, A. Corrada, G. Doddington, K. Kirchoff, M. Ordowski and B. Wheatley, "Syllable — A Promising Recognition Unit for LVCSR," *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, pp. 207-214, Santa Barbara, California, USA, December 1997.

[83] D. Kahn, *Syllable-Based Generalizations in English Phonology*, Indiana University Linguistics Club, Bloomington, Indiana, USA, 1976.

[84] S. Greenberg, "Speaking in Shorthand - A Syllable-Centric Perspective for Understanding Pronunciation Variation," *Proceedings of the ESCA Workshop on Modeling Pronunciation Variation for Automatic Speech Recognition*, Kerkrade, The Netherlands, May 3-6, 1998.

[85] J. G. Wilpon, B.-H. Juang, and L. R. Rabiner, "An investigation on the use of acoustic sub-word units for automatic speech recognition," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pp. 821-824, Dallas, USA, April 1987.

[86] K. F. Lee, "Context-dependent Phonetic Hidden Markov Models for Speaker-Independent Continuous Speech Recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 38, pp. 599-609, April 1990.

[87] R. G. Leonard, "A Database for Speaker-Independent Digit Recognition," *Proceedings of International Conference on Acoustics, Speech, and Signal Processing,* 1984.

[88] P.E. Gill, W. Murray, and M.H. Wright, *Practical Optimization,* Academic Press, New York, 1981.

[89] L.E. Baum, et. al., "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *Annals of Mathematical Statistics,* vol. 41, no. 1, pp. 164-171, 1970.

[90] K.F. Lee, *Large Vocabulary Speaker Independent Continuous Speech Recognition*, Ph. D. dissertation, Carnegie Mellon University, Pittsburgh, USA, 1988.

[91] J. Odell, *The Use of Context in Large Vocabulary Speech Recognition*, Ph. D. dissertation, Cambridge University, UK, 1997.

[92] S.J. Young and P.C. Woodland, "State Clustering in HMM-Based Continuous Speech Recognition," *Computer Speech and Language*, Vol. 8, No. 4, pp. 369-384, October 1993.

[93] A. Ganapathiraju et. al., "WS97 Syllable Team Final Report," *Proceedings of the 1997 LVCSR Summer Research Workshop*, Center for Language and Speech Processing, Johns Hopkins University, Baltimore, Maryland, USA, December 1997.

[94] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, John Wiley and Sons, New York, USA, 1991.

[95] S. Kullback, *Information Theory and Statistics,* Dover Publications, Mineola, NY, USA, 1997.

[96] S.E. Fahlman, "An empirical study of learning speed in back-propagation networks. *Technical Report CMU-CS-88-162*, Carnegie Mellon University, Pittsburgh, PA, USA, September 1988.

[97] R. Schwartz and S. Austin, "A Comparison of Several Approximate Algorithms for Finding Multiple (N-Best) Sentence Hypotheses", *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 701-704, 1991.

[98] F. K. Soong and E. F. Huang, "A Tree-Trellis Based Fast Search for Finding the N Best Sentence Hypotheses in Continuous Speech Recognition", *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 705-708, 1991.

[99] P. Baldi, "Gradient descent learning algorithm overview: A general dynamical systems perspective," *IEEE Transactions on Neural Networks*, vol. 6, pp. 182--195, 1995.

[100] S. Amari, "Theory of Adaptive Pattern Classifiers," *IEEE Transactions on Electronic Computers,* vol. 16, no. 3, pp. 299-307, 1967.

[101] S. Amari, "Mathematical Foundations of Neurocomputing," *Proceedings of the IEEE*, vol. 78, no. 9, 1990.

[102] L. Nguyen, R. Schwartz, Y. Zhao and G. Zavaliagkos, "Is N-Best Dead?", *Proceedings of the DARPA Human Language Technology Workshop*, pp. 386-388, March 1994.

[103] J. K. Chen and F. K. Soong, "An N-Best Candidates-Based Discriminative Training for Speech Recognition Applications", *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 1, pp. 206-216, January 1994.

[104] J. Tebelskis, *Speech Recognition using Neural Networks*, Ph. D. dissertation, Carnegie Mellon University, Pittsburgh, USA, 1995.

[105] A. Waibel, et. al., "Phoneme Recognition using Time-Delay Neural Networks," IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 37, pp. 328-339, March 1989.

[106] A. Robinson, "An application of recurrent nets to phone probability estimation," *IEEE Transactions on Neural Networks,* vol. 5, no. 2, pp. 298-305, 1994.

[107] A. Robinson, et. al., "The Use of Recurrent Neural Networks in Continuous Speech Recognition," in *Automatic Speech and Speaker Recognition -- Advanced Topics*, chapter 19, Kluwer Academic Publishers, 1996.

[108] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, USA, 1969.

[109] Y. Bengio, "A Connectionist Approach to Speech Recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 4, pp. 3-22, 1993.

[110] K.G. Murty and S. Kabadi, "Some NP-Complete Problems in Quadratic and Nonlinear Programming," *Mathematical Programming,* vol. 39, pp. 117-129, 1987.

[111] N. Morgan and H. Bourlard, "Continuous Speech Recognition---An introduction to the hybrid HMM/connectionist approach," *IEEE Signal Processing Magazine*, vol. 13, no. 3, pp. 24-42, May 1995.

[112] D.P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York, 1982.

[113] E.K.P. Chong and S.H. Zak, *An Introduction to Optimization*, John Wiley & Sons, Inc., New York, NY, USA, 1996.

[114] O.L. Mangasarian, *Nonlinear Programming*, McGraw-Hill Book Company, New York, NY, USA, 1969.

[115] E. Osuna, et. al. “An Improved Training Algorithm for Support Vector Machines,” *Proceedings of the IEEE NNSP’97*, pp. 24-26, Amelia Island, USA, September 1997.

[116] G. Zoutendijk, *Methods in Feasible Directions — A Study in Linear and Non-linear Programming*, Elsevier Publishing Company, New York, NY, USA, 1960.

[117] J. Platt, Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods, In *Advances in Large Margin Classifiers*, MIT Press, Cambridge, MA, USA, 1999.

[118] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Wadsworth International Group, 1984.

[119] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.

[120] W. Buntine and R. Caruana. “Introduction to IND Version 2.1 and Recursive Partitioning,” *Software Manual*, NASA Ames Research Center, Moffet Field, CA, 1992.

[121] W. Buntine, “*Tree Classification Software*,” The Third National Technology Transfer Conference and Exposition, Baltimore, MD, USA, December 1992.

[122] W. Buntine, *A Theory of Learning Classification Rules*, Ph.D. dissertation, University of Technology, Sydney, Australia, 1991.

[123] P. Sollich, “Probabilistic methods for Support Vector Machines,” *Advances in Neural Information Processing Systems 12*, MIT Press, Cambridge, MA, USA, 2000.

[124] M. Franzini, et. al., “Connectionist Viterbi Training: A New Hybrid Method for Continuous Speech Recognition,” *Proceedings of the International Conference on Acoustics, Speech and Signal Processing,* vol. 1, pp. 425-428, Albuquerque, NM, USA, 1990.

[125] P. Haffner, et. al., “Integrating Time Alignment and Neural Networks for High Performance Continuous Speech Recognition,” *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 105-108, 1991.

[126] M.E. Tipping, “The Relevance Vector Machine,” *Advances in Neural Information Processing Systems 12*, pp. 652–665, MIT Press, Cambridge, MA, USA, 2000.

[127] J. Kwok, "Moderating the Outputs of Support Vector Machine Classifiers," *IEEE Transactions on Neural Networks,* vol. 10, no. 5, September 1999.

[128] M.A. Bacchiani, *Speech Recognition System Design Based on Automatically Derived Units*, Ph. D. dissertation, Boston University, Boston, USA, 1999.

[129] J. Chang and J. Glass, "Segmentation and Modeling in Segment-based Recognition, *Proceedings of Eurospeech*, pp. 1199-1202, Rhodes, Greece, 1997.

[130] N. Strom, et. al., "Acoustic Modeling Improvements in a Segment-Based Speech Recognizer," *Proceedings of IEEE ASRU Workshop*, Keystone, CO, USA, December 1999.

[131] A.K. Halberstadt, *Heterogeneous Acoustic Measurements and Multiple Classifiers for Speech Recognition,* Ph. D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, USA, November 1998.

[132] H. Bourlard and N. Morgan, "Hybrid HMM/ANN Systems for Speech Recognition: Overview and New Research Directions," In C. L. Giles and M. Gori (Eds.), *Adaptive Processing of Sequences and Data Structures*, vol. 1387 of Lecture Notes in Artificial Intelligence, pp. 389-417, Springer-Verlag, Berlin, Germany, 1998.

[133] J. Chang, *Near-Miss Modeling: A Segment-Based Approach to Speech Recognition*, Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, 1998.

[134] J.J. Odell, V. Valtchev, P.C. Woodland and S.J. Young, "A One Pass Decoder Design for Large Vocabulary Recognition," *Proceedings of ARPA Human Language Technology Workshop,* pp. 405-410, Princeton, New Jersey, USA, March 1994.

[135] K. Seymore, et. al., "The1997 CMU Sphinx-3 English Broadcast News Transcription System," *Proceedings of the 1997 DARPA Speech Recognition Workshop*, Lansdowne, VA, USA, 1998.

[136] R. M. Schwartz and S. Austin, "Efficient, High-Performance Algorithms for N-Best Search", in *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 6-11, June 1990.

[137] D. Deterding, et. al., "Vowel Recognition", available at *http://.ics.uci.edu/pub/machine-learning-databases/undocumented/connectionist-bench/vowel/*, August 2000.

[138] J. Hamaker, A. Ganapathiraju, J. Picone and J. Godfrey, "Advances in Alphadigit Recognition Using Syllables," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 421-424, Seattle, Washington, USA, May 1998.

[139] P. Loizou and A. Spanias, "High-Performance Alphabet Recognition," *IEEE Transactions on Speech and Audio Processing*, pp. 430-445, November 1996.

[140] R. Cole et. al., "Alphadigit Corpus," *http://.cse.ogi.edu/CSLU/corpora/alphadigit*, Center for Spoken Language Understanding, Oregon Graduate Institute, 1997.

[141] Hamaker, J. et. al., "A Proposal for a Standard Partitioning of the OGI AlphaDigit Corpus," available at *http://isip.msstate.edu/projects/lvcsr/recognition_task/alphadigits/data ogi_alphadigits/trans_eval.text*.

[142] T. Hain et. al., "The 1998 HTK System for Transcription of Conversational Telephone Speech," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 57-60, Phoenix, USA, 1998.

[143] Heab-Umbach, et. al., "Acoustic Modeling in the Philips Hub-4 Continuous-Speech Recognition System," *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop,* Lansdowne, VA, USA, February 1998.

[144] R. Sundaram, J. Hamaker, and J. Picone, "TWISTER: The ISIP 2001 Conversational Speech Evaluation System," *Proceedings of the Speech Transcription Workshop*, Linthicum Heights, Maryland, USA, May 2001.

[145] J. Tenenbaum, et. al., "Separating Style and Content," in *Advances in Neural Information Processing Systems 9*, MIT Press, Cambridge, MA, USA, 1997.

[146] J. Weston, and C. Watkins, "Support Vector Machines for Multi-Class Pattern Recognition," *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, 1999.

[147] Y. Freund and R. E. Schapire, "A Short Introduction to Boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, pp. 771-780, September, 1999.

[148] C.J.C. Burges and B. Scholkopf, "Improving the Accuracy and Speed of Support Vector Machines, "*Advances in Neural Information Processing Systems-9*, Morgan Kaufman, 1997.

[149] C.J.C. Burges, "Simplified Support Vector Decision Rules," 13th International Conference on Machine Learning, pp. 71-77, 1996.

[150] R. Sundaram and J. Picone, "The Effects of Transcription Errors," *Proceedings of the Speech Transcription Workshop*, Linthicum Heights, Maryland, USA, May 2001.

[151] G. Williams and S. Renals, "Confidence Measures for Hybrid HMM/ANN Speech Recognition," *Proceedings of Eurospeech*, pp. 1955-1958, Rhodes, Greece, September 1997.

[152] S. Fine, et. al., "A Hybrid GMM/SVM Approach to Speaker Identification," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 417-420, 2001.

[153] N. D. Smith and M.J.F. Gales, "Speech Recognition using SVMs," *Neural Information Processing Systems - 2001,* Vancouver, British Columbia, Canada, 2001.

[154] S. Chakrabartty, et. al., "Hybrid Support Vector Machine / Hidden Markov Model Approach for Continuous Speech Recognition," *Proceedings of the 43rd IEEE Symposium on Circuits and Systems,* vol. 2, pp. 828-831, 2000.

[155] T. Jaakkola and D. Haussler, "Exploiting Generative Models in Discriminative Classifiers", *Advances in Neural Information Processing Systems 11*, MIT Press, Cambridge, MA, USA, 1998.

[156] N. Brunel, and J.P. Nadal, "Mutual information, Fisher Information and Population Coding," *Neural Computation,* vol. 10, pp. 1731-1757, 1998.

APPENDIX

FISHER KERNELS

This appendix describes the computation of Fisher scores which can be used as inputs to Fisher kernels which were introduced in chapter (8). We will see how these Fisher scores can be easily obtained with minor modifications to the Baum-Welch computations [19]. For simplicity we will derive the required quantities assuming single mixture Gaussian models with diagonal covariances.

Let us first define some of the terms we will use for estimates of the constituents of the Fisher score vectors. Let $\boldsymbol{O}$ be the observation sequence and let M be the models under consideration with a parameter set λ. The likelihood of the observation sequence given the model is defined as,

$$L_{\lambda}(O/M) = \log P_{\lambda}(O/M) \tag{141}$$

such that,

$$\frac{\partial L_{\lambda}}{\partial \lambda} = \frac{1}{P_{\lambda}(\boldsymbol{O}|M)} \frac{\partial}{\partial \lambda} P_{\lambda}(O/M) \tag{142}$$

The posterior probability can be written in terms of the α and β as,

$$P_{\lambda}(\boldsymbol{O}|M) = \sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_j(t)\beta_j(t).$$

In order to introduce the other parameters in the model into the above equation, we can rewrite it as,

$$P_{\lambda}(\boldsymbol{O}|M) = \sum_{t=1}^{T} \sum_{j=1}^{N} \left\{ \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} \right\} b_j(\boldsymbol{o}_t) \beta_j(t) \ . \tag{143}$$

The HMM is definition is composed of the means, variances and transition probabilities. Each of these components of the model definition contribute to the definition of the Fisher score. In the following sections we derive the contribution of these components towards the Fisher score vector for an HMM instance.

A.1. Transition Probability

The transition probabilities need to be handled carefully in order to guarantee that the transitions out of any state sum to unity. For this reason, a regularization function is used to redefine the transitions as,

$$a_{ij} = \frac{f_a(h_{ij})}{\sum_k f_a(h_{i'k})} \text{ and } f_a(x) = e^x \tag{144}$$

which is also known as softmax [(19)]. Then,

$$\frac{\partial a_{ij}}{\partial h_{ik}} = a_{ij}(\delta_{kj} - a_{ik}) . \tag{145}$$

where δ is the Kroneker delta.

Since P depends on a_{ij}'s, applying the chain rule gives,

$$\frac{\partial}{\partial h_{ik}} P_{\lambda}(O/M) = \sum_{j} \frac{\partial}{\partial a_{ij}} P_{\lambda}, (O/M) \frac{\partial a_{ij}}{\partial h_{ik}}. \tag{146}$$

Differentiating (143) with respect to a_{ij} and using it along with (145) in (146) yields,

$$\frac{\partial}{\partial h_{ik}} P_{\lambda}(O/M) = \sum\sum \alpha_i(t-1) a_{ij} (\delta_{kj} - a_{ik}) b_j(\boldsymbol{o}_t) \beta_j(t) \tag{147}$$

The above equations used in conjunction with (142) provide means to get the components in the Fisher score vector corresponding to the transition probabilities.

A.2. Mean

In order to get the contribution of the means towards the Fisher scores, we need to start with the definition of a Gaussian as in (3). Differentiating this equation with respect to the d^{th} component of the mean of the distribution corresponding to the j^{th} state, we get,

$$\frac{\partial}{\partial \mu_{jd}} b_j(\boldsymbol{o}_t) = b_j(\boldsymbol{o}_t) \left\{ \frac{o_{td} - \mu_{jd}}{\sigma^2_{jd}} \right\}. \tag{148}$$

From (143), we know that,

$$\frac{\partial}{\partial b_j(\boldsymbol{o}_t)} P_{\lambda}(O/M) = \sum_{t=1}^{T} \left\{ \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} \right\} \beta_j(t) . \tag{149}$$

Using the chain rule for partial derivatives, we get,

$$\frac{\partial}{\partial \mu_{jd}} P_{\lambda}(O/M) = \sum_{t=1}^{T} C(t,j) b_j(\boldsymbol{o}_t) \left\{ \frac{o_{td} - \mu_{jd}}{\sigma^2_{jd}} \right\}, \tag{150}$$

where,

$$C(t,j) = \left\{ \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} \right\} \beta_j(t) \; . \tag{151}$$

Using (150) and (151) in (142), we compute the contribution of the mean vectors toward the Fisher scores.

A.3. Variance

As mentioned earlier, for simplicity we assume diagonal covariances in these derivations. In the case of diagonal covariances, we need to constrain the values to be positive. In order to convert the constrained set to an unconstrained set (as we did with the transitions), we use the following regularization:

$$\sigma_{jd}^{\;2} = f(z_{jd}) \tag{152}$$

and

$$f(x) = e^{x} \; . \tag{153}$$

The gradient of the output distribution with respect to the variance can then be written as,

$$\frac{\partial}{\partial \sigma^2_{jd}} b_j(\boldsymbol{o}_t) = b_j(\boldsymbol{o}_t) \frac{1}{2} \left\{ \frac{(o_{td} - \mu_{jd})^2}{(\sigma^2_{jd})^2} - \frac{1}{\sigma^2_{jd}} \right\} \; . \tag{154}$$

Using (152) and (153), we can convert (154) in terms of the regularization variable z as,

$$\frac{\partial}{\partial z_{jd}} b_j(\boldsymbol{o}_t) = b_j(\boldsymbol{o}_t)\frac{1}{2}\left\{\frac{(o_{td}-\mu_{jd})^2}{\sigma^2_{jd}} - 1\right\}. \tag{155}$$

Using (149) and the chain rule for partial derivatives, we get

$$\frac{\partial}{\partial z_{jd}} P_\lambda(O/M) = \sum_{t=1}^{T} C(t,j) b_j(\boldsymbol{o}_t)\left\{\frac{(o_{td}-\mu_{jd})^2}{(\sigma^2_{jd})^2} - 1\right\}. \tag{156}$$

By substituting (156) into (142), we can compute the contribution of the variances towards the Fisher score.